TIMEX COMPUTER FDD3000

Service Manual

TC TIMEX COMPUTER

FDD3000

SERVICE MANUAL

FOR DOUBLE
&
SINGLE DRIVE SYSTEM

REV. Ø3

CONTENTS

APPENDIX

INTRODUCTION

I) INTRODUCTION

It is not the objective of this manual how to teach how to repair but to give some technical guidelines allowing fault finding and isolation of the defect and whenever possible to solve the fault.

The troubleshooter needs to use its technical knowledge on a rational way. He will need patience, a continuity tester a multimeter and an osciloscope. As a very basic rule these are the tools required to identify and troubleshoot the faults.

Special attention is required to the non conformities with the product specification.

This manual was conceived to completely cover the product and the project, and is the most updated one.

The way this manual was written will give a progressive knowledgment of the product as you will read through it allowing you to solve all the situations.

WHAT IS THE FDD 3000

II - HARDWARE

II-2 THE OPERATING SYSTEM

The addition of the floppy disc drive system to the zx Spectrum or its compatibles TC2048 and TC2068 creates a very powerfull system, the TOS (TIMEX OPERATING SYSTEM). This OPERATING SYSTEM was developped by TIMEX giving the user a capacity from the BASIC that he never thought about.

The secret is the controller that commands the drive operation, another one for external communications that an inimaginable flexibility to TOS.

The controller has 64k Bytes of DINAMIC memory.

The input and output of data is controlled by a Z80A microprocessor at 4 MHZ.

When a TOS command is issued it passes through a paralell bus to the controller that executes the instructions, without over-loading the computer memory, since the the controller on its one is a stand alone computer.

The controller and the operating system where developped around the need to communicate. So we can find 2 RS 232 (V24) ports, which can be configured to any type of communication protocol, which means that besides the power to communicate with other users of TOS it is also possible to TALK with users of others computer systems. Other peripherals like telephonic MODEMS, PRINTERS and other devices connectable to a communication port can be driven by the FDD system.

II-3) THE MEDIA (Diskettes or simply discs)

The diskettes (Discs) are manufactured with very thin flexible material coated with a magnetic material, enclosed inside a hard plastic case. A shutter mecanically actuated protects the media when out of the floppy dics drive unit.

This protection opens automatically when the disc is inserted in the drive and when it is completely inside the READ/WRITE head thouches the disc.

The disc is of double density type allowing a formatted capacity of 160 K per side (total 320 K Bytes per disc).

Although the discs are very reliable certain basic percautions have to be taken:

- Do not expose the discs to very strong or direct sources of heat or light.
- Do not submit disc to high temperatures.
- Do not expose the disc to any type of magnetic field.
- Do not expose to dust.
- Keep discs away of humide environments.
- Do not open the protection shutter and/or thouch the media with the fingers or any other object.

The information stored in the disc can be protected in two possible ways: hardware or software. the Hardware protection is achieved trough the protection tabs, that cover or open a hole near the corners of the disc plastic enclosure, which allows or stops the write operations on the disc.

OPEN HOLE.....WRITE PROTECTED DISC

... About CP/M

CP/M means "Control Program Monitor" and was one of the first Operating Systems written in the begining of the 70' s.
Dr. Gary Kindall is considered as the father of CP/M, he was developing Software for the INTEL microprocessor 8080.

With the development of different types and sizes of floppy disc drives the need for a system that would interconnect or make all of them was created. This is one of the reasons of CP/M flexibility and transportability from one system to another.
CP/M Operating Sysrem consists off 3 parts, to know:

BIOS (BASIC INPUT/OUTPUT SYSTEM)

BDOS (BASIC DISC OPERATING SYSTEM)

CCP (CONSOLE COMAND PROCESSOR)

the BIOS is the only part of the Operating System that depends on the Hardware where CP/M is going to run. Is the only part of the Operating System that can be costumised by the Hardware manufacturer to have mind particularities of the Floppy Disc Drives, Discs, Terminals, and Printers that he produces.

BDOS can not be rellocated or modified since it is legally protected. It is the part of the Operating System that controlls all the Drives and is independent of the Hardware.

The CCP interfaces the user with the software since it translates the keyboard instrutions into instructions that the microprocessor can understand. It is also legally protected and can not be relocated or modified.

When CP/M is loaded into computer memory it is done in a well organized way occupying segments of memory. The BIOS is stored in the top of the memory.

The CCP stays immediatly below and CCP occupies the bottom part of the memory pile. The remaining memory space is reserved for the application programs and is called TPA (TRANSIT PROGRAM AREA).

The amount of memory used by the BIOS depends on the Hardware configuration and is typically of 3K Bytes.

The BDOS as a fixed length of 3.5 K Bytes. This happens because when CP/M transfer information from the RAM to the Disc, or Vice-versa does it always in blocks of 128 Bytes up to a maximum of 16 drives. (only for maximum on the case of the TIMEX FDD). The physical operation between the DISCS and the RAM is handled by the BIOS.

Below the BIOS and BDOS location in memory we find the CCP and the TPA, the lowest part of memory is formed by a block of 256 K Bytes the contains the system variables.

When the Computer is Switch-On, the first operation performed is to load the CP/M Operation System, from is location in the Disc, to the RAM of the Computer. This is called "BOOTSTRAPPING" or simply system "BOOT". Deppending on the system this can be achieved trough a special combination of keystrokes on the keyboard, in the case of the FDD 3000 this happens automatically after the insertion of the disc into the floppy disc drive. We know that this operation is finished when the screen displays the log-on message from the Hardware manufacturer followed by the promt:

A>

This symbol indicates that the CCP is functioning and is waiting to answer any command line.

If by any reason it is necessary re-initialize CP/M after the power-on of the computer, it is possible to execute a "WARM-BOOT" warm reset of the Operating System. This is some times required after a system error and the subsequent Operating System "error message". This procedure loads BDOS and CCP into their memory location meaning that the computer is ready to accept new commands again.

CP/M is a registered trademark of DIGITAL RESEARCH.

Some passages of "USING CP/M" by PETER GOSLING where quoted above.

OPERATING PROCEDURES

III- OPERATION PROCEDURES.

In this charter we will through the installation procedure of the Floppy Disc Drive unit.

The FDD 3000 supports two different Operating systems:

TOS - TIMEX OPERATING SYSTEM

CP/M - CONTROL PROGRAM MONITOR

The Hardware configuration depends directly of the Operating System to be used. Lets see why:

If we want to use TOS software we need the following Hardware:

- MICROCOMPUTER (Z80 based)
- TV SET OR MONITOR
- INTERFACE FOR COMMUNICATION
- FLOPPY DISC DRIVE SYSTEM FDD 3000
- PRINTER (RS232) (OPTIONAL)

In this configuration the interface changes according to the computer to be used.

If your request is to run Software in CP/M format to approaches can be used, one using a computer as console and the other is to use a TERMINAL 3000, in this case the configuration will be:

- TERMINAL 3000
- MONITOR
- FLOPPY DISC DRIVE SYSTEM FDD 3000
- PRINTER (RS232 SERIE) (OPTIONAL)

III-2) THE CONNECTIONS BY SEQUENCE

1- Connect the curly cable from the FDD 3000 to the right communication interface for your computer.

2- Carefully engage the interface connector of the interface in the computer edge-connector.

3- Connect the RS232 cable for the printer into the choosen channel "A" or "B".

4- If you have a ZX SPECTRUM you can use a monitor, just connect the monitor RCA plug from the monitor into the socket at the back of the FDD 3000 system also marked "MONITOR". If you have a ZX SPECTRUM and not a MONITOR you have to use the TV set, connecting the TV cable to the appropriate plug on the computer. If you any other TIMEX computer and a MONITOR you have to plug the RCA jack into your computer, socket marked with MONITOR.
It is completelly impossible to connect the tv cable to the video plug at the back of the FDD 3000 since no picture will be produced at all in the TV screen.
If you are using a TERMINAL 3000 to work with this system the monitor cable RCA must be connected to the MONITOR plug at the back of the FDD 3000 system.

5- Connect all the power supplies to the mains socket 220 VOLT. If you are usin the ZX SPECTRUM or TC computers connect the plugs from the power supplies to the correct sockets at the back pannels of the computers. Connect the TV set or the MONITOR to the mains.

6- Them power-on of:

- POWER ON TV SET OR MONITOR
- POWER ON OF THE FDD 3000
- POWER ON OF THE COMPUTER
- POWER ON OF THE PRINTER

HOW TO TEST THE FDD 3000

IV- HOW TO TEST THE FDD 3000

Configure the system to operate in the TOS mode:

1- POWER ON THE TV SET OR THE MONITOR

2- POWER ON THE FDD 3000

3- POWER ON THE PRINTER

4- POWER ON THE COMPUTER

5- PUT THE TOS FORMATTED DISC INTO DRIVE A AND THE CP/M DISC IN DRIVE B

6- DO RESET ON THE FDD 3000 CONTROLLER RESET BOTTON

7- DO RESET ON THE COMPUTER INTERFACE BY PRESSING RESET BOTTON ON THE INTERFACE

ON THE SCREEN THE FOLLOWING MESSAGE WILL APPEAR AFTER A SECOND

C 1982 SINCLAIR RESEARCH LTD
C 1984 TIMEX - TOS VA.2

9- THE SCREEN WILL CLEAR AND THE FOLLOWING MESSAGE WILL BE DISPLAYED

"DAMAGE'S CHANNEL'S" (if there is no RS 232 communicatio between channel "A" and "B")

If the communication is possible between the two channels the following message is shown:

" TAKE OUT THE PLUG RS 232 &
CONNECT THE PRINTER CORD! "

PRESS C TO CONTINUE

9- PRESS KEY C,
THE SCREN WILL THEN SHOW:

" MONITOR EMULATOR FOR TC2048/TC2068
Copyright TMX PORTUGAL
VERSION V A1.1 11, APRIL 1986

PLEASE PUT THE CP/M DISKETTE IN DRIVE A:
AND PRESS THE CCONTROLLER RESET BUTTON: "

10- REMOVE THE DISC IN DRIVE "A". TURN TO SIDE B AND INTRODUCE IT IN DRIVE "A" AGAIN. PRESS THE RESET BUTTON ON THE CONTROLLER: THE SCREEN WILL SHOW:

```
CP/M VERSION 2.2
COPYRIGHT DIGITAL RESEARCH INC!

CBIOS VERSION VA1.1 COPYRIGHT TMX PORTUGAL

3, JUNE 1986
```

11- TYPE "T" + "RETURN"
THE SCREEN DISPLAYS:

```
TESTE DO FDD 3000
PREPARADO POR F.FARIA & Z.SILVA
C 1986 TMX PORTUGAL
```

TESTE DE ACESSO A DRIVE A CONSTITUIDO POR:

1- CRIACAO DO FICHEIRO "TEMP.TMP"
2- ESCRITA THE 128 K BYTES (ASCII"?") NO FICHEIRO
3- FECHO DO FICHEIRO
4- ABERTURA DE FICHEIRO
5- LEITURA DO FICHEIRO
6- APAGAR AO FICHEIRO

TESTE DO CANAL (:CH_A) DO TIMEX FDD 3000

VERIFIQUE SE A SEQUENCIA DE CARACTERES ESTA CORRECTA!

1234567890QWERTYUIOPASDFGHJKLZXCVBNNM

TESTE DE MEMORIA DO TIMEX FDD 3000

MEMORIA OK !!!

A>

12- CONNECT THE "RCA" PLUG AND VERIFY IF THE SCREEN IMAGE IS CORRECT.

IF THE ANSWER IS yes THE CONTROLLER IS APPROVED !!!

IMPORTANT ! IMPORTANT ! IMPORTANT ! IMPORTANT !

BEFORE STARTING THE REPAIR SESSION PLEASE MAKE SHURE
THAT YOU HAVE BACKUP COPIES OF THE TEST SOFTWARE

IMPORTANT ! IMPORTANT ! IMPORTANT ! IMPORTANT !

ELECTRICAL SCHEMATICS

V - ELECTRICAL SCHEMATICS

LIST OF SIGNALS:

1 - CLOCK 16 MHZ (XIN) (PIN 45 GA121)

2 - CLOCK 16 MHZ (XOUT) (PIN 44 GA121)

3 - CLOCK 4 MHZ (XOUT) (PIN 42 GA121)

4 - CLOCK 8 MHZ (XOUT) (PIN 42 GA121)

5 - CLOCK 1.8432 MHZ (X1) (PIN 31 WD2123 - U10)

6 - CLOCK 1.8432 MHZ (X2) (PIN 32 WD2123 - U10)

7 - D10A/D10K (PIN 15,16 - U17)

8 - DRCV/SWC 8 (PIN 15,16 - U17)

9 - BASE Q1 (D43C1)

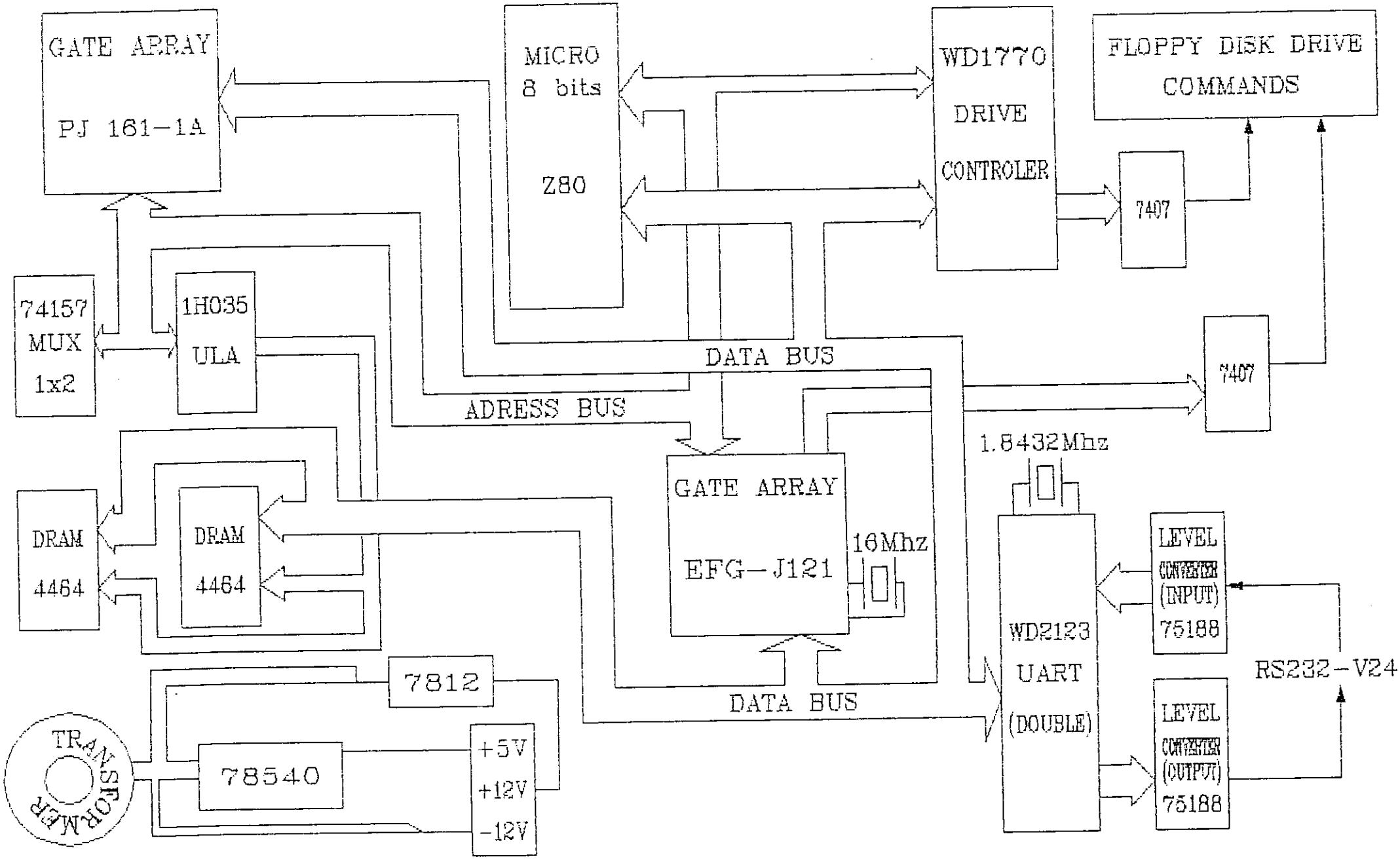
10- TINC (PIN 12 - U17)

11- C1 (PIN 10 - U17)

12- COLLECTOR Q1 (D43C1)

13- VOUT (U16)

14- VAC


GATE ARRAY
PJ 161-1A
MICRO
8 bits
Z80
WD1770
DRIVE
CONTROLER
FLOPPY DISK DRIVE
COMMANDS
7407
74157
MUX
1x2
1H035
ULA
DATA BUS
ADRESS BUS
7407
1.8432Mhz
GATE ARRAY
EFG-J121
16Mhz
DRAM
4464
DRAM
4464
WD2123
UART
(DOUBLE)
LEVEL
(INPUT)
75188
RS232-V24
LEVEL
(OUTPUT)
75188
DATA BUS
7812
TRANSFORMER
78540
+5V
+12V
-12V

TROUBLESHOOTING

VI - TROUBLESHOOTING

Since the FDD 3000 is composed of severall blocks of Hardware and Software, it is necessary to fully identify the origin of the fault if it exists.

A good rule to troubleshoot is to always use a disc that you are absolutely shure that is 100 % working.

Connect all the different parts of the FDD 3000 according to what was described in chapter III as follows:

Power On the TV set or the MONITOR, switch on the FDD 3000 and computer.

Insert the System Disc into drice "A".

Verify if the power LED is ON:

If the LED in the FDD 3000 is OFF preform the operations described above in the reverse order.

Verify if the FUSE located in the rear pannel of the FDD conduts.

Verify if all the plugs are fully engaged, well inserted, and are in goood conditions.

Verify if all the terminals are well crimped and inserted in the terminals and housings.

The LED on the Floppy Disc Drive remains ON but the systems fully operates, i.e. responds correctly to the commands

Verify if the line "IN USE" is not interrupted.

Make shure that the DIODE that connects the "IN USE" signal and ground is good.

Remember that this signal is active LOW meaning a very low voltage level near 0 V (0.4 v).

Verify that the signals in U8 change.

Verify if PIN 11 of the gatte array GA 121 is not short circuited.

Note: Configure the system to work in CP/M mode and load the the Operating System. In good circunstances the LED located in the drive blinks when loading the Operating System, in the end it should stay in the OFF stage. The Led would lit again only in case of a disc access. The "IN USE" signal is set ON or OFF after the drive selection preformed by the Operating System.

The LED remains always ON and the drive can only recognize the TRACK 00.

If the system is still configured to run CP/M verify if the communication cable with the console or computer is in perfect conditions (all the lines warrantying galvanic connection between plugs).

Do a system RESET checking if the system is able to locate other tracks then only TRACK 00. If that happens the problem is located at the CONSOLE/COMPUTER level.

After power on nothing happens:

Verify if the address lines of the controller are not stuck to a fix logic value.

Verify if the DATA lines are active changing logic values.

Verify if the PINS of RD and WR of the processor (Z80)are accessed (PINS 21 and 22 of U1).

The system loads TOS but not the CP/M Operating System:

Verify if the SELECT lines of the different MEMORY banks are correctly selected by the signals "ASEL", "CP/M3", "CP/M2", "CP/M1" (PINS 22, 23, 24, 25, of gatte array GA121).

Note: After POWER ON the bank automaticaly selected is the one corresponding to "ASEL".

Have in mind that for a correct selection of the MEMORY banks only one can be selected at the time (EXOR).

To check if the MEMORY selects are correct procede as follows: after insering the CP/M disc in drive "A" check, with the help of an Osciloscope, PIN 22 of gatte array GA 121. If it must go LOW when trying to load the Operating System. Try to load the Operating System by pressing thr RESET button on the controller, all the lines should stay HIGH.

It does not communicate with the interface although it loads TOS.

Verify if PIN 21 of gatte array GA121 (MODE) is correctly selected in J1B and J1A. This PIN selects the type of communication to be performed (BYTES or NIBLES).

Verify if all the DATA lines DO (DATA OUT) oscilate within the specified limits.

Verify if the DI lines (DATA IN) oscilate inside the correct values.

After POWER ON nothing happens.

Verify if the system is powered.

Verify if there is no excess power consumption in any of the lines.

Verify the clock signal between PINS 44 and 45 of the GA 121.

Verify the 8 MHZ signal at PIN 42 of gatte array GA 121.

Verify if the contacts of the RESET button are free from any short-circuit.

Verify if the RESET line (PIN 34) of the gatte array GA 121 is at the logic state "1" HIGH.
Press the RESET button and check if the signal across PIN 34 of the same gatte array varies inside the correct values.

Verify if the line " INT" is HIGH. Analyse the response of this signal with changes in the RESET.

Verify if the "HALT" command is activated.

Verify if the ON/OFF swicht is operational cutting or conducting when actuated.

Check if the soldered points between the the circuit and the secondary terminals of the transformer are in good conditions.

Check the primary of the transformer.

Measure the DC value of the SPSU output in the open circuit situation.

With the help of resistive loads on the secondary power lines check the voltage drop.

Verify that the power LED is in good conditions and the connecting wires also assure a reliable connection.

Impossible to FORMAT a disc in drive "A".

Verify if the line "WD" is assuring a good connection between the drive and the PCB.

Check if the LOGIC level of PIN 22 of WD 1770 varies during the format operation.

Verify the inverter U-7 (Gatte IN-11, Gatte OUT-10).

Even with the Hardware protection (protection TAB of the disc - open) it is possible to write in the disc.

Verify if the line "WRPT" assures the connection between the drive and PIN 10 of U-11 (GA 121).

Verify the contact of pin of U-11 with the PCB.

The controller outputs to the RS 232 line but does not receive through the same line.

Verify if the +12V and -12V voltages are applied to the BUFFER's CONVERTERS U9 (1489 or 75189).

Check if the decoupling capacitors on the ENABLE pins are correctly inserted and in good working conditions.

Check if the ripple measured in the power lines are inside resonable limits.

Verify if the clock signal that drives the communication has no other interfering signal in top of it.

Produce an artificial shunt between the input and output pins of the converter, with a very simple software program open a file sending after ASCII characters through the channel. this helpsto check the efficiency of the level converters.

After RESET the Controller reads TRACK 00 and the following 3 TRACKS

Verify if any line of U-3 (ULA 1h035) is not stuck LOW.

Check if U-3 do not heat.

Verify the address lines of the microprocessor before and after the multiplex circuit.

Make shure that A0 and A1 change value allowing the execution of the first 4 TOS instructions addressed to the floppy disc drive.

A0	A1		A0	A1	
0	0	DS	0	0	TK00
0	1	MD	0	1	IP
1	0	DIR	1	0	RD
1	1	STEP	1	1	WPTR
TRANSMIT			RECEIPT		

Verify if the SELECT lines of the MEMORY banks of the gatte array GA121, after RESET are accessed. Check the status of PIN 9 of U-3 (256CS).

Verify if the Z80 line INT is active.

Verify PIN 28 (WD 1770 - INT).

Verify the oscillations in PIN 27 of Z80 M1 (Machine Code Cycle).

The characters sent to the printer do not to the ones displayed on the screen.

Verify if the printer is configured in the same way as the controller.

- BAUD RATE
- PARITTY
- STOP BITS
- TEXT/BYTES
- BITS/CHARACTER

Check the ripple of the power supply are not out of the specified.

Check if there are no spurious signals on top of the clock signal of the communication IC.

Verify if the UART clock signal is stable.

Check if the decoupling capacitors of the level converters are in good conditions.

Check all the supply voltages at the communication converters.

Loads TOS incompletely

Verify if the address lines multiplexed in U4 and U5 are stuck and do not oscilate.

Does not load TOS and does not select the drive.

Check if the multiplexer MUX U2 (74157) is commutating the address A4 and A15.

Verify if the control signal MPX PIN 14 of U3 (1H035) is not stuck LOW by GA121 or U3.

No VIDEO output

Check if the pins of the RCA plug are well soldered.

Assure that the grouding of the cable is good.

Verify the connections of the curly cable.

Does not execute RESET because the RESET line is hold LOW.

Verify if C16 is not LOW.

Check if the PINS of the RESET Button are not short-circuited.

MECHANICAL FAULTS

APPENDIX

The Operating System when invoced by mistake will produce an error message, which could be:

HARDWARE FAULT ON DISK

This message will be display whenever a READ or WRITE error occur during access to the disc.

The error can be located in the disc drive Hardaware or by the disc itself. If the disc in drive A is not present this error may in some circunstances be generated. Note that the system can not operate without a disc in drive A.

A faulty disc can also generate this message.

DISC CORRUPTED

This message may be displayed if a disc is partially destroid due to magnetic fields, heat etc.

The Operating System loaded in the controller memory is corrupted by noise generated by electrical interference in the power line.

The new disc inserted in drive A has a new version of the Operating System or no Operating System was previuosly saved in the disc. In any of those re-BOOT the system.

If the message ramains after checking al the above is because the disc is fault. try to FORMAT it again but before try to transfer the files to another disc which will work in the majority of the cases.

----- IS WRITE PROTECTED

Appears after a trial to erase a file or directory that is protected via the instruction "ATTR * " (sotware protection).
This error may also be generated by "SAVE *", "ERASE *","MOVE *" and "OPEN *".

DISK--------IS SOFTWARE W/P

It is not possible to create, update, erase, rename or even write protect a file or directory in a software protected disc with the instruction "ATTR *" this error may be generated by the following instructions: "SAVE *", "OPE *", "CLOSE* ", "DIM *", "ERASE *", "LET* ", "MOVE*", "PRINT*", or "ATTR *". Note that the instruction FORMAT does not produce this error since it formats the disc even if the same is software write protected.

DISC------IS HARDWARE W/P

The disc is Hardware protected by the use of the protection TABS.

The instruction FORMAT * (DISC) does not format a Hardware write protected disc but generates the error message "HARDWARE FAULT ON DISC ----------".

DIRECTORY FULL ON DISC-----

Generated whenever the systems tries to update a file however there is no space in the disc or directory. This error may occur with the following instructions "DIM *", "CLOSE *", "MOVE *" and "SAVE *".

A drive directory accepts up to 128 files entry. The disc itself uses one entry for its own name. Any other directory uses one entry. The files occupy one entry for each 16k length. This means that a file with 7k uses only one entry and a file with 50k uses 4 .

ILLEGAL USE OF A ROOT NAME

it is impossible to specify a name for the root or from the root. This error may be generated by the instructions that have a PATHNAME as argument; "GOTO *", "GOSUB *" AND "CAT *" (AT THE ROOT LEVEL), also "DIM *" , "ERASE *", "SAVE *", (FROM THE ROOT).

CHANNEL BUSY

The instruction "OPEN *" generates this an error message when trying to specify this channel. Change channel or firts close the OPEN file.

CHANNEL NOT OPEN

The instructions that generate this error message are "CLOSE *", "LIST *", "PRINT *", "INPUT *" or "RESTORE *" when trying to use a channel not previously OPEN.

ILLEGAL DRIVE NAME

The instruction FORMAT * generates this error if the specified drive is wrong.

ILLEGAL DISC NAME

The instruction FORMAT * generates this error message when a non valid name is specified for the drive in the second ALFANUMERIC entry. It is necessary to specify a disc name or directory and not a PATHNAME.

DISC --------FULL

The disc as no free space, this error is generated by the instructions "SAVE *", "MOVE *" OR "PRINT *".

THIS ARE THE TOS ERROR MASSAGES

APPENDIX II-1 RS 232 PIN-OUT

CONNECTIONS REQUIRED FOR HTE RS 232 PROTOCOL RS 232C (V24).

PLUG FEMMALE TYPE "D" 9 PINS

PIN NUMBER	PIN TYPE
01	NC
02	TX DATA
03	RX DATA
04	DTR
05	CTS
06	NC
07	GND
08	NC
09	+12V

THE ABOVE PIN-OUT CORRESPONDS TO THE FDD PLUGS TO CONNECT TO CHANNEL A OR CHANNEL B.

NOTE: TX DATA >>>>>>>>>>> TRANSMIT DATA

RX DATA >>>>>>>>>>> RECEIVE DATA

DTR >>>>>>>>>>>>>>> "DATA TERMINAL READY" ready to receive information.

CTS >>>>>>>>>>>>>>> "CLEAR TO SEND"

PLUG 25 PINS MALE D TYPE

03	.....	RX DATA
04	.....	TX DATA
05	.....	CTS
20	.....	DTR
07	.....	GND
06	.....	DSR (+12 V)

NOTE: DSR >>>>>>>>>>> "DATA SET READY"

PARTS LIST

SPARE PARTS LIST

ITEM	PART NAME	P/NUMBER	%
0005	DISKETTE, TOS VA.2	398-917020	5%
0040	DISKETTE, CP/VM VA.2	398-909464	5
0170	DRIVE, HITACHI HFD350SX	394-970100	1
0180	FUSE, SLOW BLOW 250V/0A5	396-920033	10
0275	ASSY, CONN, CENTER END 21 WAYS	301-970110	5
0285	CORD, WITH MOULD"D" PLUG FEM 15WAYS	395-930035	2
0295	SUB. ASSY, PUSH BOTTOM	301-975110	8
0405	CONN.34 POSITIONS EDGE CARD	395-930055	1
0410	CABLE, FLAT 34 COND. 28AWG, GRAY	395-930040	2
0440	PLUG, HOUSING, 4 POS	396-920050	3
0460	CONN.M PIN HEADER 2X17 PINS DBL ROW	301-946004	2
0465	PLUG, PCB MOUNTING, 4 WAYS	395-930069	5
0470	PLUG, RGT.ANG. PCB MOUNTING 4 WAYS	395-930070	5
0490	CONN."D" TYPE 9 PINS SOCKET DGR MALE	301-940200	5
0505	CONN. LOCK HD.STRAIGHT POS 21 WAYS	395-930084	5
0510	XTAL. 16MHZ, HC-18/U	395-930060	2
0515	XTAL, 1.8432MHZ, HC-18U	395-930066	2
0580	IC, CPU MK3880-Z80	330-910901	5
0590	IC,WD117U	395-930010	5
0595	IC,WD2123	395-930015	5
0600	IC, 75188	395-930085	3
0605	IC, 75189	395-930090	3
0635	IC, 74LS273	395-930025	3
0645	IC, DRAW 4464-150NS	335-912115	5
0650	IC, ULA 11035	334-812014	3
0811	SUB ASSY-PCB	301-975130	
0812	IC, EFGJ121	395-930105	5
0813	IC, EFGJ161	395-930065	5

LISTING (MACRO)CbIOS

```
;       .Z80
;*****************************************************
;                       BIOS
;               Versao para densidade dupla
;
;               VERSAO FINAL EM : 85/10/29
;               ACTUALIZACAO : 3/6/86
;
; BY : Antonio Nobrega
;
; Esta versao substitui a versao de 20/1/83. O novo
; BIOS tem processamento de erro para recuperacao
; de problemas de hardware.
; Emendado a nao detecao de erro de read a segunda tentativa
;
;*****************************************************
;
;       CP/M REFERENCE CONSTANTS
;
;
IMAG    EQU     100H            ;offset para a REL -1 ou 0
MSIZE   EQU     20              ;Capacidade da memoria em K bytes
 IAS    EQU     (MSIZE-20)*1024+IMAG    ;Deslocamento em relacao aos
CCP     EQU     BIAS            ;20 K bytes
BDOS    EQU     CCP+806H
CBIOS   EQU     CCP+1600H
;
IOBYTE  EQU     0003H   ;Estrutura do IOBYTE nao implementado
CDISK   EQU     0004H   ;Drive corrente
NDRIVE  EQU       4     ;Numero de drives previsto
NTENT   EQU      10     ;Numero de tentativas em caso de erro
                        ;no acesso a disco
SECLOG  EQU     128     ;Sector logico
SECFIS  EQU     256     ;Sector fisico
SEC1    EQU       0     ;Sector fisico representado em 1 byte
;
;               Caracteres ASCII
;
LF      EQU     0AH     ;Line feed
CR      EQU     0DH     ;Carriage return
FFEED   EQU     0CH     ;Form feed
EOF     EQU     1AH     ;End of file
ESCAPE  EQU     1BH
;
;               Protocolo comunicacoes CONIN CONOUT
;
ASTAT   EQU     0
BSTAT   EQU     40H
CSTAT   EQU     10H
DSTAT   EQU     50H
STMSK   EQU     DSTAT
COM     EQU     2FH     ;Endereco do porto de commicacoes
SERVICO EQU     0FH
MASK    EQU     5FH
BYTEON  EQU     0AH
STANDBY EQU     05H
WSTA    EQU     06H     ;Flag indica conin esta espera caractere
;
;*****************************************************
;*                      Descricao do  Hardware       *
;*****************************************************
;
;
;       comunicacao serie
;       =================
```

```
;
;       Chip WD 2123
;       ============
;
;       Enderecos:

CANALA  EQU     80H             ;Endereco do porto A
SIOASTS EQU     CANALA + 1      ;Endereco do registo de status
SIOACMD EQU     CANALA + 1      ;Endereco do registo de comando
SIOADAT EQU     CANALA          ;Endereco do registo de data
SIOABR  EQU     10H             ;Endereco do registo de baud/rate

ANALB   EQU     40H             ;Endereco do porto B
SIOBSTS EQU     CANALB+1        ;Endereco do registo de status
SIOBCMD EQU     CANALB+1        ;Endereco do registo de comando
IOBDAT  EQU     CANALB          ;Endereco do registo de data
SIOBBR  EQU     11H             ;Endereco do registo de baud/rate
;
;       Comandos de programacao

ACOM    EQU     00000111B       ;input e output enable
;
;       Flags do registo de status
;
RXRDY   EQU     00000001B       ;Receiver READY
RR      EQU     01111000B       ;Erro na recepcao ou na emissao
IOERR   EQU     37H             ;Palavra de comando com reset flag
SIOWORD EQU     27H             ;palavra de comando com RQT="0"
PTS     EQU     20H
IOMASK  EQU     81H
;
;
;       Controlador de disco
;       ====================
;
;
;       Chip    FD1770
;       ==============
;
;       Enderecos:

FDC     EQU     0C0H
FDCSTS  EQU     FDC             ;Reg de stat
FDCMD   EQU     FDC             ;Reg de comando
FDCTRK  EQU     FDC + 1         ;Reg da pista
FDCSEC  EQU     FDC + 2         ;Reg de sector
FDCDAT  EQU     FDC + 3         ;Reg de dados

;       Comandos de programacao
;
RSTCMD  EQU     05H     ;Restore
SEKCMD  EQU     15H     ;Seek c/ verify
STICMD  EQU     55H     ;Step in
STOCMD  EQU     75H     ;Step out
RDSCMD  EQU     80H     ;Read sector
RDMCMD  EQU     98H     ;Read multiple sectors
WRSCMD  EQU     0A0H    ;Write sector c/ precompensacao
RDACMD  EQU     0CCH    ;Read address
FICMD   EQU     0D0H    ;Force interrutp
;
;       Flags do status

BUSY    EQU     0       ;Busy
DLOST   EQU     2       ;Data lost
RFOUND  EQU     4       ;Record not found
WPROTE  EQU     6       ;Write proctect
```

```
;       Outros registos
;       ===============
;
DRQ     EQU     02FH    ;Registo de comunicacoes
                        ;     bit 0-6  usados comunicacao em nibbles
                        ;     bit 7    DRQ
HRD     EQU     0E0H    ;Registo de controle
                        ;     bit 0 - drive sel 0
                        ;     bit 1 - drive sel 1
                        ;     bit 2 - drive sel 2
                        ;     bit 3 - drive sel 3
                        ;     bit 4 - side select
                        ;     bit 5 - /DDEN ( densidade)
                        ;     bit 6 - /BOOT
                        ;     bit 7 - IN USE
;
;       Programacao
;
DENSID  EQU     01011111B       ;Programacao do porto de controle
INTPRG  EQU     11011111B
MSKDRV  EQU     00001111B       ;Mascara para os drives
LUZ     EQU       7             ;Acende luz no drive seleccionado
;
;
;*****************************************************************
;*                      Tabela de JUMPs do BIOS                  *
;*****************************************************************
;
        JP      BOOT
        JP      WBOOT
        JP      CONST
        JP      CONIN
        JP      CONOUT
        JP      IMPR
        JP      PUNCH
        JP      READER
        JP      HOME
        JP      SELDSK
        JP      SELTRK
        JP      SETSEC
        JP      SETDMA
        JP      READ
        JP      WRITE
        JP      IMPRST
        JP      SECTRAN
;
;
; LOGO inicial da versao do bios
;
SIGNON:         DB      ESCAPE
        DEFM    "H"
        DB      ESCAPE
        DEFM    "J"
        DEFM    "CP/M Version 2.2"
        DB      CR,LF
        DEFM    "Copyright by DIGITAL RESEARCH, Inc."
        DB      CR,LF,LF
        DEFM    "Cbios Version A1.1 Copyright by THX PORTUGAL"
        DB      CR,LF,LF
        DEFM    "3, June 1986"
        DB      CR,LF,LF
        DEFM    "$"
;
;
;*****************************************************************
;*                      Subrotinas do BIOS                       *
```

```
;**********************************************************
;
;
;       ***************************************************
;       *                      BOOT                       *
;       ***************************************************
;
                ;Procedimento a ser executado depois do LOADER
                ;em que inicializa o IOBYTE o disco corrente a
                ;zero, e afixa a mensagem inicial. Por fim ini-
                ;cializa a page 0 do CP/M com os valores conve-
                ;nientes passando em seguida o comando ao CCP.
                ;
BOOT::  XOR     A
        LD      (IOBYTE),A      ;Reset IOBYTE
        LD      (CDISK),A       ;Inicializa o drive selecionado
        LD      (UNIT),A
        CALL    INIT
        CALL    INITRS  ;inicializa canais rs_232
        LD      HL,SIGNON
        CALL    PMSG            ;Imprimir a mensagem inicial
        JP      GOCPM
;
;
;       ***************************************************
;       *                     WBOOT                       *
;       ***************************************************
;
                ;Inicializa o hardware, carrega o CCP e o BDOS
                ;e inicialza a pagina 0 do CP/M passando em se-
                ;guida o comando ao CCP.
;
WBOOT:: LD      SP,STACK
        DI
        CALL    INIT
        XOR     A
        CALL    LIGHT
        LD      HL,CCP          ;Endereco de base do CP/M
        LD      C,FDCDAT
        LD      B,0
        LD      D,1
WBLOP:: LD      A,D
        CALL    RDLOOP
        JR      NZ,WBOOT
        LD      A,(0H
        INC     D
        CP      D
        JR      NZ,WBLOP
        CALL    STEPIN          ;pista 1
        XOR     A
        CALL    RDLOOP  ;Sector 0 pista 1
        CALL    LIGHTOF
GOCPM:: LD      A,0C3H
        LD      (00H),A
        LD      HL,CBIOS+3
        LD      (01H),HL
        LD      (05H),A
        LD      HL,BDOS
        LD      (06H),HL
        XOR     A
        LD      (BLREAD),A      ;Nada no BUFFER de blocagem
        LD      (BLALT),A
        LD      BC,0080H
        CALL    SETDMA
        LD      A,(CDISK)
        LD      C,A
```

```
        JP      CCP
;;
;
; ***********************************************************************
; *               Subrotinas de I/O de caracteres                       *
; ***********************************************************************
;
;
;       ***************************************************
;       *                     CONST                       *
;       ***************************************************
;
;
                        ;Indica o status da consola
                        ;   .parametros de saida
                        ;       reg. A - FF se existe caracter
                        ;              - 00 se nao existe caractere
                        ;   .altera o resgisto A
CONST:: IN      A,(COM)
        AND     MASK
        CP      04
        JR      Z,CONSON
        XOR     A
        RET
CONSON::        LD      A,0FFH
        RET
;
;
;       ***************************************************
;       *                     CONIN                       *
;       ***************************************************
;
;
CONIN:: LD      (SVSTK),SP
        LD      SP,STACK
        CALL    RXBYTE  ;Recebe byte do terminal
        LD      C,A
        LD      SP,(SVSTK)
        RET
;
;
;       ***************************************************
;       *                     CONOUT                      *
;       ***************************************************
;
;
CONOUT::        LD      (SVSTK),SP
        LD      SP,STACK
        LD      A,C
        CALL    TXBYTE  ;Transmite byte
        LD      SP,(SVSTK)
        RET
;
;
;       ***************************************************
;       *                     IMPR                        *
;       ***************************************************
;
;
IMPR::  LD      (SVSTK),SP
        LD      SP,STACK
        CALL    PRINTER
        LD      SP,(SVSTK)
        RET
;
;
```

```
;       ************************************************************
;       *                        IMPRST                            *
;       ************************************************************
;
;
IMPRST::        LD      (SVSTK),SP
        LD      SP,STACK
        CALL    PRINSTS
        LD      SP,(SVSTK)
        RET
;
;
;       ************************************************************
;       *                        READER                            *
;       ************************************************************
;
;
READER::        LD      (SVSTK),SP
        LD      SP,STACK
        CALL    RDR
        LD      SP,(SVSTK)
        RET
;
;       ************************************************************
;       *                        PUNCH                             *
;       ************************************************************
;
;
PUNCH:: LD      (SVSTK),SP
        LD      SP,STACK
        CALL    PUN
        LD      SP,(SVSTK)
        RET
;
;
; ************************************************************************
; *              Subrotinas de I/O de disco                              *
; ************************************************************************
;
;       ************************************************************
;       *                        SECTRAN                           *
;       ************************************************************
;
;
SECTRAN::       LD      A,C             ;Determina se sao os 128 bytes me
        AND     01H             ;significativos ou os mais do sector
        LD      (HIGH1),A       ;a ler
        LD      A,C
        RRA
        LD      C,A     ;Determina o sector logico de 256 bytes
                        ;a ser lido
        EX      DE,HL
        ADD     HL,BC
        LD      L,(HL)
        LD      H,0
        RET
;
;
;       ************************************************************
;       *                        SELTRK                            *
;       ************************************************************
;
;
SELTRK::        LD      A,C
```

```
        LD      (TRACK),A
        RET
;
;
;
;       ********************************************************
;       *                     SETSEC                           *
;       ********************************************************
;
;
SETSEC::        LD      A,C
        DEC     A
        LD      (SECTOR),A
        RET
;
;
;       ********************************************************
;       *                     SETDMA                           *
;       ********************************************************
;
;
SETDMA::        LD      (POINTR),BC
        RET
;
;       ********************************************************
;       *                     SELDSK                           *
;       ********************************************************
;
;
SELDSK::        LD      HL,0
        LD      A,C
        CP      NDRIVE
        RET     NC
        LD      (UNITDR),A
        LD      L,A
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL
        LD      DE,DPHTAB
        ADD     HL,DE
        RET
;
;       ********************************************************
;       *                     HOME                             *
;       ********************************************************
;
;
HOME::  LD      A,(UNITDR)
        LD      C,A
        CALL    LIGHT
        LD      A,RSTCMD
        OUT     (FDCMD),A
        CALL    TEMP
HOMLP:: IN      A,(FDCSTS)
        BIT     BUSY,A
        JR      NZ,HOMLP
        CALL    LIGHTOF
        XOR     A
        RET
;
;
;       ********************************************************
;       *                     READ                             *
```

```
        ;*****************************************************

EAD::   LD      (SVSTK),SP
        LD      SP,STACK
        CALL    SAVJMP  ;guarda salto de int e coloca novo jmp
        LD      A,(BLREAD)      ;Testar se o BUFFER ja foi
        AND     A       ;lido alguma vez
        JR      Z,RDCONT
        CALL    IGUAL   ;Testa se os sectores no BUFFER
                        ;e para ser lido sao iguais
                        ;se SIM
                        ;       entao A = 0
                        ;       senao A = FF
        AND     A
        JR      Z,REXISTE
        LD      A,(BLALT)       ;Testa se o sector no BUFFER
                        ;foi alterado
                        ;se SIM
                        ;       entao "blalt" = FF
                        ;       senao "blalt" = 0
        AND     A
        JR      Z,RDCONT
        CALL    WR256
        AND     A
        JR      NZ,RDFIM
CONT::          LD      HL,BLDRIV
        LD      A,(UNITDR)
        LD      (HL),A          ;Bldriv <-- UNITdr
        INC     HL
        LD      A,(TRACK)
        LD      (HL),A          ;Blpist <-- track
        INC     HL
        LD      A,(SECTOR)
        LD      (HL),A          ;Blsect <-- sector
        INC     HL
        XOR     A
        LD      (HL),A          ;Blalt <-- 0 (* nao alterado *)
        DEC     A
        LD      (BLREAD),A
        CALL    RD256
        AND     A
        JR      NZ,RDFIM        ;Erro
        LD      (BLALT),A
REXISTE::       CALL    RTRANSF
        XOR     A
DFIM::  LD      (ERRORD),A      ;flag de erro de leitura
        CALL    PUTJMP  ;repoi salto
        LD      SP,(SVSTK)
        RET

;
        ;*****************************************************
        ;                       WRITE                       *
        ;*****************************************************
;

JRITE:: LD      (SVSTK),SP
        LD      SP,STACK
        CALL    SAVJMP  ;guarda 38h
        PUSH    BC      ;Salvar se e acesso a directoria
                        ;se e o primeiro ou se e um acesso
                        ;normal
        LD      A,(BLREAD)      ;Testa se o BUFFER ja foi
        AND     A               ;lido alguma vez
        JR      Z,WRCONT
```

```
        CALL    IGUAL     ;Testa se os sectores no BUFFER
                          ;e para ser lido sao iguais
                          ;se SIM
                          ;       entao A = 0
                          ;       senao A = FF
        AND     A
        JR      Z,WEXISTE
        LD      A,(BLALT)         ;Testa se o sector no BUFFER
                                  ;foi alterado se SIM
                                  ;       entao "blalt" = FF
                                  ;       senao "blalt" = 0
        AND     A
        JR      Z,WRCONT
        CALL    WR256
        AND     A
        JR      NZ,WRPFIM
WRCONT::        LD      HL,BLDRIV
        LD      A,(UNITDR)
        LD      (HL),A            ;Bldriv <-- UNITdr
        INC     HL
        LD      A,(TRACK)
        LD      (HL),A            ;Blpist <-- track
        INC     HL
        LD      A,(SECTOR)
        LD      (HL),A            ;Blsect <-- sector
        INC     HL
        XOR     A
        LD      (HL),A            ;Blalt <-- 0 (* nao alterado *)
        DEC     A
        LD      (BLREAD),A
        CALL    RD256
        AND     A
        JR      NZ,WRPFIM         ;Erro
WEXISTE::       CALL    WTRANSF
        POP     BC
        LD      A,C
        CP      1
        JR      NZ,WRNDIR
        CALL    WR256
        AND     A
        JR      NZ,WRFIM
        LD      (BLREAD),A
WRNDIR::        XOR     A
        JR      WRFIM
WRPFIM::        POP     BC
WRFIM:: CALL    PUTJMP
        LD      SP,(SVSTK)
        RET
;
;
;       ************************************************************
;       *                       OUTRAS                             *
;       ************************************************************
;
;
        ;       *** init ***
        ;
        ;Inicializa todo o hardware
        ;       .controlador de disco
;
;
INIT::  LD      A,INTPRG
        OUT     (HRD),A
        LD      B,NDRIVE
        LD      A,0FEH
INTLP:: OUT     (HRD),A
```

```
          RLC       A
          DJNZ      INTLP
          LD        A,DENSID
          OUT       (HRD),A
          AND       0FEH
          OUT       (HRD),A
          LD        (IMAGSEL),A
          LD        A,RSTCMD
          OUT       (FDCMD),A
          CALL      TEMP
INITLP::              IN        A,(FDCSTS)
          BIT       BUSY,A
          JR        NZ,INITLP
          XOR       A
          LD        (BLREAD),A
          LD        (BLALT),A
          LD        (UNIT),A
          DEC       A
          LD        B,NDRIVE
          LD        HL,DRVTBL
INLP::    LD        (HL),A
          INC       HL
          DJNZ      INLP
          CALL      LIGHTOF
          RET
;
;
          ;         *** RD256 ***
          ;
          ;Le 256 bytes para o BUFFER
;
;
RD256::   LD        A,(BLDRIV)
          LD        C,A
          CALL      LIGHT
          LD        A,NTENT
          LD        (RETRY),A
          CALL      PREPARA ;tenta dez vezes o seek
          AND       A
          JR        NZ,RDERR
RTRY::    LD        HL,BLBUFF         ;endereco buffer
          CALL      RDDSK    ;tenta leitura
          AND       A
          JR        Z,RFIM   ;se o ok sai
          LD        A,(RETRY)
          DEC       A
          LD        (RETRY),A
          JR        NZ,RTRY
RDERR::   LD        B,255
RDOUT::   CALL      LIGHTOF
          LD        A,B
          RET
RFIM::    LD        B,00
          JR        RDOUT
RDDSK::   IM        1
          EI
          LD        A,RDSCMD
          OUT       (FDCMD),A
RLP::     IN        A,(DRQ)
          RLA
          JR        NC,RLP
          INI
          JR        RLP
;
;
          ;         *** WR256 ***
```

```
;
;
WR256::  LD      A,(BLDRIV)      ;Acesso ao drive do buffer
         LD      C,A
         CALL    LIGHT
         LD      A,NTENT
         LD      (RETRY),A       ;10 tentativas
         CALL    PREPARA ;tenta escrita
         AND     A       ;erro?
         JR      NZ,WRERR        ;sim jump sai
WRTRY::  LD      HL,BLBUFF
         CALL    WRDSK   ;tenta escrita
         AND     A       ;erro?
         JR      Z,WFIM
         BIT     6,A     ;disco protegido?
         CALL    NZ,DSKPRT       ;sim logo para consola
         CP      "R"     ;retry?
         JR      Z,WRTRY ;sim tenta nova escrita
         LD      A,(RETRY)       ;tenta dez vezes
         DEC     A
         LD      (RETRY),A
         JR      NZ,WRTRY
WRERR::  LD      B,255   ;flag erro
WROUT::  CALL    LIGHTOF ;desliga led
         LD      A,B
         RET
WFIM::   LD      B,0     ;flag write ok
         JR      WROUT
WRDSK::  EI
         LD      C,FDCDAT        ;endereco porto data
         LD      A,WRSCMD
         OUT     (FDCMD),A
WRLP::   IN      A,(DRQ)
         RLA
         JR      NC,WRLP
         OUTI
         JR      WRLP
;
;
         ;       *** PRINTER ***
         ;
         ;Transmite byte recebido registo C
         ;para canal serie seleccionado
         ;
PRINTER::        LD      A,SIOWORD
         OUT     (SIOACMD),A
         CALL    ERROA
IMPHR::  IN      A,(SIOASTS)
         BIT     7,A     ;cts low ?
         JR      Z,IMPHR ;nao espera
         BIT     0,A     ;registo de tx. esta ready?
         JR      Z,IMPHR ;nao espera
IMPTX::  LD      A,C
         OUT     (SIOADAT),A
         RET
;
;
;
;
                 ;       *** PRINSTS ***
                 ;
                 ;Da o estado do periferico (rs232)
                 ;associado com o canal seleccionado
;
;
PRINSTS::        CALL    ERROA
```

```
          IN        A,(SIOASTS)
          BIT       7,A
          JR        NZ,LPTON
          XOR       A
          RET
 PTON::   LD        A,0FFH
          RET
;
:

                    ;         *** PUN: ***
                    ;
                    ;Out do caractere pelo canal B
                    ;
PUN::     LD        A,SIOWORD
          OUT       (SIOBCMD),A
 UNCTS::            IN        A,(SIOBSTS)
          AND       81H       ;espera cts low e thr empty
          CP        81H
          JR        NZ,PUNCTS
          LD        A,C
          OUT       (SIOBDAT),A
          RET

;
:
                    ;         *** RDR: ***
                    ;
                    ;Le caractere do canal B
 DR::     LD        A,SIOWORD
          OUT       (SIOBCMD),A       ;forca CTS low
RDRLP::   IN        A,(SIOBSTS)       ;get status
          AND       7FH       ;limpa cts
          CP        8         ;testa erro
          JR        NC,RDRERR
          BIT       RXRDY,A ;byte ready?
          JR        Z,RDRLP ;espera loop
          IN        A,(SIOBDAT)       ;le byte
          LD        C,A
          LD        A,ACOM  ;forca CTS a high
          OUT       (SIOBCMD),A
          LD        A,C
          RET
 RDRERR::           LD        A,37H
          OUT       (SIOBCMD),A
          JR        RDRLP


          ;         *** WTRANSF ***
          ;
          ;Transfer 128 bytes do endereco POINTR
          ;para o BUFFER  - high ou low.
;

 TRANSF::           LD        HL,BLBUFF
          LD        BC,SECLOG
          LD        A,(HIGH1)
          AND       A
          JR        Z,WLOW
          ADD       HL,BC
 LOW::    EX        DE,HL
          LD        HL,(POINTR)
          LDIR
          XOR       A
          DEC       A
          LD        (BLALT),A         ;PROVISORIO
```

```
        RET
;
;
        ;       *** rtransf ***
        ;
        ;Transfer 128 bytes do BUFFER -high ou low-
        ; para o POINTR.
;
;
RTRANSF::       LD      HL,BLBUFF
        LD      BC,SECLOG
        LD      A,(HIGH1)
        AND     A
        JR      Z,RLOW
        ADD     HL,BC
RLOW::  LD      DE,(POINTR)
        LDIR
        RET
;
;
        ;       *** igual ***
        ;
        ;Testa se o sector que esta no BUFFER e
        ;igual ao que se quer ler ou escrever
;
;
IGUAL:: PUSH    IX
        LD      IX,BLDRIV
        LD      A,(UNITDR)
        CP      (IX+0)
        JR      NZ,DIFER
        LD      A,(TRACK)
        CP      (IX+1)
        JR      NZ,DIFER
        LD      A,(SECTOR)
        CP      (IX+2)
        JR      NZ,DIFER
        LD      A,(ERRORD)      ;ve se houve erro no read anterior
        AND     A
        JR      NZ,DIFER
        POP     IX
        RET
DIFER:: LD      A,0FFH
        POP     IX
        RET
;
;Entry: B=Sector Count, C=Starting Sector, HL=Start Address
;Reads every other sector (1 sector interleve)
                        ;
                        ;       *** RDLOOP ***
;
RDLOOP::        OUT     (FDCSEC),A
        LD      A,RDSCMD
        OUT     (FDCMD),A
RLLP::  IN      A,(DRQ)
        RLA
        JR      NC,RLLP
        INI
        JR      NZ,RLLP
RELLP:: IN      A,(FDCSTS)
        BIT     BUSY,A
        JR      NZ,RELLP
        AND     1CH
        RET
;
;
```

```
;
;
;       *** point ***
;
;Devolve no reg HL o apontador do
;descritor do drive especificado
;no reg. A
;
;
POINT:: LD      HL,DRVTBL
        AND     A
PNT1::  RET     Z
        INC     HL
        DEC     A
        JR      PNT1
;
;
;       *** setpin ***
;
;Faz avancar a cabeca do drive de uma
;posicao
;
;
STEPIN::        LD      A,STICMD
        OUT     (FDCMD),A
        CALL    TEMP
SETPLP::        IN      A,(FDCSTS)
        BIT     BUSY,A
        JR      NZ,SETPLP
        RET
;
;
;       *** light ***
;
;Seleciona o drive e acende a luz
;do mesmo.
;
;
LIGHT:: LD      C,A
        LD      (TUNIT),A
        LD      A,(IMAGSEL)
        RES     LUZ,A
        OR      MSKDRV
        LD      (IMAGSEL),A
        OUT     (HRD),A
        LD      A,(UNIT)
        CALL    POINT
        IN      A,(FDCTRK)
        LD      (HL),A
        LD      B,0FEH
LIGCNT::        DEC     C
        JP      M,LIGEND
        RLC     B
        JR      LIGCNT
LIGEND::        LD      A,(IMAGSEL)
        AND     B
        OUT     (HRD),A
        LD      (IMAGSEL),A
        LD      A,(TUNIT)
        LD      (UNIT),A
        CALL    POINT
        LD      A,(HL)
        CP      0FFH
        JR      NZ,LIGNRST
        LD      A,RSTCMD
        OUT     (FDCMD),A
```

```
        CALL    TEMP
LIGLP:: IN      A,(FDCSTS)
        BIT     BUSY,A
        JR      NZ,LIGLP
        XOR     A
LIGNRST::       OUT     (FDCTRK),A
        RET
;
;
        ;
        ;       *** Inirs ***
        ;
        ;Inicializa os canais com os parametros
        ;dados pela tabela SIOTAB
        ;
INITRS::        LD      HL,SIOTAB       ;Inicializa HL com ADD tabela
        CALL    PUTMOD  ;Constroi a palavra de modo
        OUT     (SIOACMD),A
        LD      A,ACOM
        OUT     (SIOACMD),A     ;Palavra de comando
        LD      A,(HL)
        OUT     (SIOABR),A      ;Baud rate
        ;
        ;Canal A programada
        ;
        INC     HL      ;Inicializa hl com parametros canal B
        CALL    PUTMOD
        OUT     (SIOBCMD),A
        LD      A,ACOM
        OUT     (SIOBCMD),A
        LD      A,(HL)
        OUT     (SIOBBR),A
        RET
;
;
        ;
        ;       *** Putmod ***
        ;
        ;Forma a palavra de modo
        ;
PUTMOD::        LD      A,(HL)  ;Baud rate factor
        INC     HL
        ADD     A,(HL)  ;Bits/char
        INC     HL
        ADD     A,(HL)  ;Parity enable
        INC     HL
        ADD     A,(HL)  ;Parity select
        INC     HL
        ADD     A,(HL)  ;Stop bits
        INC     HL      ;Endereca baud rate
        RET
;
;
;
        ;       *** lightoff ***
        ;
        ;A apaga a luz do drive selecionado
;
;
LIGHTOF::       LD      A,(IMAGSEL)
        SET     LUZ,A
        LD      (IMAGSEL),A
        OR      MSKDRV
        OUT     (HRD),A
        LD      A,(IMAGSEL)
        OUT     (HRD),A
```

```
        RES     LUZ,A
        LD      (IMAGSEL),A
        RET
;
;
;
;
        ;       ***TXBYTE***
        ;Transmite um byte para a consola
;
;
TXBYTE::        PUSH    DE
        PUSH    BC
        LD      (CHATMP),A      ;Guarda caractere temporario
        LD      B,A     ;Caractere em A
TX0::   CALL    TXBY
        JR      Z,TXOUT
TX1::   IN      A,(COM)
        AND     5FH
        CP      07
        JR      NZ,TX1
        LD      A,(CHATMP)
        LD      B,A
        JR      TX0
TXOUT:: XOR     A
        POP     DE
        POP     BC
        RET
;
;
TXBY::  LD      A,B
        AND     0FH
        LD      E,A     ;E Fica com 1.o nibble
        LD      A,B
        AND     0F0H
        RRCA
        RRCA
        RRCA
        RRCA
        LD      D,A     ;D Fica com 2.o nibble
READY:: LD      A,BYTEON
        OUT     (COM),A
        IN      A,(COM)
        AND     MASK    ;Deixa passar bits 6,4,3,2,1,0
        CP      SERVICO
        JR      NZ,READY
        LD      A,E
        OR      CSTAT
        OUT     (COM),A ;Tx. 1.o nibble e bit 4 a "1"
        LD      B,A
        LD      C,0
TXTW1:: IN      A,(COM) ;Espera eco
        DEC     C
        JR      Z,TXERR
        AND     MASK    ;Limpa bits 7 e 5
        CP      B
        JR      NZ,TXTW1        ;eco diferente espera
        LD      A,D
        OR      BSTAT   ;Set flag
        OUT     (COM),A ;TX 2.o nibble e bit 6 a "1"
        LD      B,A
        LD      C,0
TXTW2:: IN      A,(COM) ;Espera eco
        DEC     C
        JR      Z,TXERR
        AND     MASK
```

```
          CP        B
          JR        NZ,TXTW2
          LD        A,STMSK  ;Envia fim de caractere
          OUT       (COM),A
          XOR       A
          RET
TXERR::   LD        A,07H
          OUT       (COM),A
          AND       A         ;SET FLAG ERRO
          RET
;
;
          ;         ***RXBYTE***
          ;Recebe um byte
          ;
;
;
RXBYTE::            PUSH      BC
          PUSH      DE
RX0::     CALL      RXBY
          JR        Z,RXOUT  ;Byte ok sai
RX1::     IN        A,(COM)
          AND       5FH
          CP        7
          JR        Z,RX1
          JR        RX0
RXOUT::   LD        A,B       ;COLOCA CARACTERE EM A
          POP       DE
          POP       BC
          RET
;
;
;
RXBY::    LD        A,WSTA   ;Flag estou espera byte
          OUT       (COM),A
RXTW0::   IN        A,(COM)
          AND       MASK
          CP        STANDBY
          JR        NZ,RXBY
          LD        A,0FH
          OUT       (COM),A  ;Transmite flag ready
          LD        C,0
RXTW1::   IN        A,(COM)
          DEC       C
          JR        Z,RXERR
          LD        E,A
          AND       STMSK
          CP        CSTAT
          JR        NZ,RXTW1
          LD        A,E
          OUT       (COM),A  ;Transmite eco
          AND       0FH
          LD        E,A       ;Guarda 1.o nibble
          LD        C,0
RXTW2::   IN        A,(COM)
          DEC       C
          JR        Z,RXERR
          BIT       6,A       ;Testa se e 2.o nibble
          JR        Z,RXTW2
          OUT       (COM),A  ;Transmite eco
          AND       0FH
          RLCA
          RLCA
          RLCA
          RLCA
          OR        E
```

```
          AND       7FH
          LD        B,A       ;Guarda caractere
RXTW3::   IN        A,(COM)
          AND       5FH
          CP        7
          JR        Z,RXERR
          CP        STMSK
          JR        NZ,RXTW3
          LD        A,STMSK ;fora de servico
          OUT       (COM),A
          XOR       A
          RET
RXERR::   LD        A,07H
          OUT       (COM),A
          AND       A
          RET
          ;         *** temp ***
          ;
          ;Temporizador de 50 uS
;
;
TEMP::    PUSH      BC
          LD        B,18
          DJNZ      $
          POP       BC
          RET
;
;
          ;         *** seek ***
          ;
          ;Posiciona a cabeca do drive selecionado na
          ;pista especificada na posicao de memoria
          ;"track".
;
;
SEEK::    PUSH      BC
          LD        A,NTENT
          LD        (TENTAT),A        ;n.o de tentativas de seek
          IN        A,(FDCTRK)
          LD        B,A
          LD        A,(BLPIST)
          CP        B
          JR        Z,SEKOK           ;ja la estava !!
SEEKLP::            LD        A,(BLPIST)
          OUT       (FDCDAT),A
          CALL      SEKINT  ;tenta seek
          AND       10111111B         ;mascara bit write protected
          AND       A         ;erro?
          JR        Z,SEKOK ;nao jump
          CALL      HOME      ;forca restor
          LD        A,(TENTAT)        ;n.o de tentativas.
          DEC       A
          LD        (TENTAT),A
          JR        NZ,SEEKLP         ;ate dez tentativas
          INC       A         ;set erro seek
          JR        SEKERR
SEKOK::   XOR       A
SEKERR::            POP       BC
          RET
SEKINT::            LD        A,SEKCMD
          IM        1
          EI
          OUT       (FDCMD),A         ;Comando de ir para a pista
                                      ;selecionada
          JR        $         ;espera saida interrupt
;
```

```
;
;       *** prepara ***
;
;Prepara a leitura ou a escrita de
;um sector. Para isso actualiza o reg.
;e sector posiciona a cabeca na pista
;e prepara os registos
;
;
PREPARA::       LD      A,(BLSECT)
        OUT     (FDCSEC),A
        LD      C,FDCDAT        ;Endereco do reg. de dados
        JP      SEEK            ;Posiciona a cabeca
;
ERRDSK::        LD      HL,ABORT
        CALL    PMSG
ERRAGN::        CALL    RXBYTE
        RES     5,A
        PUSH    AF
        CALL    TXBYTE
        POP     AF
        CP      "A"
        JP      Z,WBOOT
        CP      "I"
        RET     Z
        CP      "R"
        JR      NZ,ERRAGN
        RET
;
; DSKPRT : coloca msg erro diskette protegida
;
DSKPRT::        LD      HL,MSGPRO
        CALL    PMSG
        JR      ERRDSK
;
; TIMOUT : timeout do pun, e printer
;
;
; SAVJMP : guarda salto do int e coloca novo salto
;
SAVJMP::        PUSH    AF
        PUSH    BC
        LD      HL,38H
        LD      DE,BUFINT
        LD      BC,0003
        LDIR
        DEC     HL
        LD      DE,INTRPT
        LD      A,D
        LD      (HL),A
        LD      A,E
        DEC     HL
        LD      (HL),A
        DEC     HL
        LD      (HL),0C3H       ;jmp
        POP     BC
        POP     AF
        RET
;
; PUTJMP : repoi salto 38h
;
PUTJMP::        PUSH    AF      ;salva reg
        LD      DE,38H
        LD      HL,BUFINT
        LD      BC,0003
        LDIR
```

```
        POP     AF
        RET
;
; INTRPT : devolve subrotina status do disco
;
INTRPT::        POP     AF      ;destroi ret
        IN      A,(FDCSTS)      ;le status
        AND     5CH     ;mascara bits erro
        RET
;
;
        ;       *** pmsg ***
        ;
        ;Imprime uma string que e apontada pelo
        ;par HL e termina em $.
;
;
PMSG::  LD      A,(HL)
        LD      C,A
        CP      "$"
        INC     HL
        RET     Z
        CALL    TXBYTE  ;transmite log cpm
        JR      PMSG
ERROA:: IN      A,(SIOASTS)
        AND     ERR
        RET     Z
        LD      A,37H
        OUT     (SIOACMD),A
        RET
ERROB:: IN      A,(SIOBSTS)
        AND     ERR
        RET     Z
        LD      A,37H
        OUT     (SIOBCMD),A
        RET
;
;
; ********************************************************************
; *          Estruturas de dados, tabelas e mensagens                *
; ********************************************************************
;       **********************************************
;       *                   MENSAGENS                *
;       **********************************************
;
;
MSGPRO::        DB      ESCAPE,"Y",55,32
        DB      LF,LF
        DEFM    "Diskette hardware protected$"
;
ABORT:: DB      CR,LF
        DEFM    "Abort, Ignore, Retry?$"
;
;       **********************************************
;       *                   Tabelas                  *
;       **********************************************
;
;
;
SECTB0::        DB      01,06,11,16,05,10,15,04
        DB      09,14,03,08,13,02,07,12
;
;
;       DISK PARAMETER BLOCK FOR STANDARD 3" FLOPPY
;               Densidade dupla
;
```

```
DPBLK0::  DW      32
          DB      3
          DB      7
          DB      0
          DW      151
          DW      63
          DB      11110000B
          DB      00000000B
          DW      16
          DW      2
;
;
;         DISK PARAMETER HEADERS FOR A 4 DISK SYSTEM
;
DPHTAB::          DW      SECTB0,0000H      ;DPH FOR UNIT 0
          DW      0000H,0000H
          DW      DIRBUF,DPBLK0
          DW      CHK0,ALL0
;
          DW      SECTB0,0000H      ;DPH FOR UNIT 1
          DW      0000H,0000H
          DW      DIRBUF,DPBLK0
          DW      CHK1,ALL1
;
          DW      SECTB0,0000H      ;DPH FOR UNIT 2
          DW      0000H,0000H
          DW      DIRBUF,DPBLK0
          DW      CHK2,ALL2
;
          DW      SECTB0,0000H      ;DPH FOR UNIT 3
          DW      0000H,0000H
          DW    DIRBUF,DPBLK0
          DW      CHK3,ALL3
;
;
;         ***************************************************
;         *               Estruturas de dados               *
;         ***************************************************
;
;
SIOTAB::          DB      02,0CH,00,00,80H,08      ;Parametros canais RS-232
                  DB      02,0CH,00,00,80H,08
BUFINT::          DS      3         ;buffer que guarda 3 bytes 38h
RETRY::  DS       1         ;Numero de tentativas de leitura/escrita
TENTAT::          DS      1         ;n.o de tentativas de seek
IMAGSEL:: DS      1         ;Imagem do registo de controle
HIGH1::  DS       1         ;Determ. sector HIGH ou LOW
DRVTBL:  DS       4         ;Tabela com a imagem dos drives
UNIT::   DS       1
TUNIT::  DS       1
UNITDR::          DS      1         ;Drive selecionado
TRACK::  DS       1         ;Pista selecionada
SECTOR::          DS      1         ;Sector selecionado
POINTR::          DS      2         ;Endereco selecionado
BLDRIV::          DS      1         ;Estrutura de dados para a blocagem/
BLPIST::          DS      1         ;Desblocagem
BLSECT::          DS      1
BLREAD::          DS      1
BLALT::           DS      1
BLBUFF::          DS      SECFIS
          DS      32        ;Stack
STACK::  DS       1
SVSTK::  DS       2         ;Variavel para armazenar o SP
CHATMP::          DS      1
;
;
```

```
;
;       ***********************************************
;       * Estruturas de dados do BDOS definidas no BIOS *
;       ***********************************************
;
;
DIRBUF::        DS      128             ;SCRATCH DIRECTORY BUFFER
ALL0::  DS      32                      ;UNIT 0 ALLOCATION BUFFER
CHK0::  DS      16                      ;UNIT 0 CHECK VECTOR
ALL1::  DS      32                      ;UNIT 1 ALLOCATION VECTOR
CHK1::  DS      16                      ;UNIT 1 CHECK VECTOR
ALL2::  DS      32                      ;UNIT 2 ALLOCATION VECTOR
CHK2::  DS      16                      ;UNIT 2 CHECK VECTOR
ALL3::  DS      32                      ;UNIT 3 ALLOCATION VECTOR
CHK3::  DS      16                      ;UNIT 3 CHECK VECTOR
ERRORD::        DS      1               ;FLAG DE READ READ ERRROR
;
        END
```

APENDIX VI

LISTING (MACRO) BIOS

```
;       MDS-800 I/O Drivers for CP/M 2.2
        (four drive single density version)

;       Version 2.2 February, 1980
;
vers    equ     22      ;version 2.2
;
;       Copyright (c) 1980
;       Digital Research
;       Box 579, Pacific Grove
;       California, 93950
;
;
true    equ     0ffffh  ;value of "true"
false   equ     not true        ;"false"
test    equ     false   ;true if test bios
;
        if      test
bias    equ     03400h  ;base of CCP in test system
        endif
        if      not test
bias    equ     0000h   ;generate relocatable cp/m system
        endif
;
patch   equ     1600h

        org     patch
cpmb    equ     $-patch ;base of cpm console processor
bdos    equ     806h+cpmb       ;basic dos (resident portion)
cpml    equ     $-cpmb  ;length (in bytes) of cpm system
nsects  equ     cpml/128        ;number of sectors to load
offset  equ     2       ;number of disk tracks used by cp/m
cdisk   equ     0004h   ;address of last logged disk on warm start
buff    equ     0080h   ;default buffer address
retry   equ     10      ;max retries on disk i/o before error
;
;       perform following functions
;       boot    cold start
;       wboot   warm start (save i/o byte)
;       (boot and wboot are the same for mds)
;       const   console status
;               reg-a = 00 if no character ready
;               reg-a = ff if character ready
;       conin   console character in (result in reg-a)
;       conout  console character out (char in reg-c)
;       list    list out (char in reg-c)
;       punch   punch out (char in reg-c)
;       reader  paper tape reader in (result to reg-a)
;       home    move to track 00
;
;       (the following calls set-up the io parameter block for the
;       mds, which is used to perform subsequent reads and writes)
;       seldsk  select disk given by reg-c (0,1,2...)
;       settrk  set track address (0,...76) for subsequent read/write
;       setsec  set sector address (1,...,26) for subsequent read/write
;       setdma  set subsequent dma address (initially 80h)
;
;       (read and write assume previous calls to set up the io parameters)
;       read    read track/sector to preset dma address
;       write   write track/sector from preset dma address
;
;       jump vector for indivisual routines
        jmp     boot
wboote: jmp     wboot
        jmp     const
        jmp     conin
```

```
        jmp     conout
        jmp     list
        jmp     punch
        jmp     reader
        jmp     home
        jmp     seldsk
        jmp     settrk
        jmp     setsec
        jmp     setdma
        jmp     read
        jmp     write
        jmp     listst  ;list status
        jmp     sectran

        maclib  diskdef ;load the disk definition library
        disks   4       ;four disks
        diskdef 0,1,26,6,1024,243,64,64,offset
        diskdef 1,0
        diskdef 2,0
        diskdef 3,0
        endef occurs at end of assembly

        end of controller - independent code, the remaining subroutines
        are tailored to the particular operating environment, and must
        be altered for any system which differs from the intel mds.

        the following code assumes the mds monitor exists at 0f800h
        and uses the i/o subroutines within the monitor

        we also assume the mds system has four disk drives
evrt    equ     0fdh    ;interrupt revert port
ntc     equ     0fch    ;interrupt mask port
 on     equ     0f3h    ;interrupt control port
 te     equ     0111$1110b      ;enable rst 0(warm boot), rst 7 (monitor)

        mds monitor equates
 n80    equ     0f800h  ;mds monitor
mon80   equ     0ff0fh  ;restart mon80 (boot error)
i       equ     0f803h  ;console character to reg-a
        equ     0f806h  ;reader in to reg-a
        equ     0f809h  ;console char from c to console out
o       equ     0f80ch  ;punch char from c to punch device
 -      equ     0f80fh  ;list from c to list device
 .      equ     0f812h  ;console status 00/ff to register a

        disk ports and commands
 se     equ     78h     ;base of disk command io ports
_tat    equ     base    ;disk status (input)
type    equ     base+1  ;result type (input)
yte     equ     base+3  ;result byte (input)

low     equ     base+1  ;iopb low address (output)
high    equ     base+2  ;iopb high address (output)

adf     equ     4h      ;read function
ritf    equ     6h      ;write function
 cal    equ     3h      ;recalibrate drive
 rdy    equ     4h      ;i/o finished mask
r       equ     0dh     ;carriage return
f       equ     0ah     ;line feed

gnon:   ;signon message: xxk cp/m vers y.y
        db      cr,lf,lf
        if      test
        db      '32'    ;32k example bios
        endif
```

```
        if      not test
        db      '00'    ;memory size filled by relocator
        endif
        db      'k CP/M vers '
        db      vers/10+'0','.',vers mod 10+'0'
        db      cr,lf,0

oot:    ;print signon message and go to ccp
        (note: mds boot initialized iobyte at 0003h)
        lxi     sp,buff+80h
        lxi     h,signon
        call    prmsg   ;print message
        xra     a       ;clear accumulator
        sta     cdisk   ;set initially to disk a
        jmp     gocpm   ;go to cp/m


boot:;  loader on track 0, sector 1, which will be skipped for warm
        read cp/m from disk - assuming there is a 128 byte cold start
        start.

        lxi     sp,buff ;using dma - thus 80 thru ff available for stack

        mvi     c,retry ;max retries
        push    b
 oot0:  ;enter here on error retries
        lxi     b,cpmb  ;set dma address to start of disk system
        call    setdma
        mvi     c,0     ;boot from drive 0
        call    seldsk
        mvi     c,0
        call    settrk  ;start with track 0
        mvi     c,2     ;start reading sector 2
        call    setsec

        read sectors, count nsects to zero
        pop     b       ;10-error count
        mvi     b,nsects
dsec:   ;read next sector
        push    b       ;save sector count
        call    read
        jnz     booterr ;retry if errors occur
        lhld    iod     ;increment dma address
        lxi     d,128   ;sector size
        dad     d       ;incremented dma address in hl
        mov     b,h
        mov     c,l     ;ready for call to set dma
        call    setdma
        lda     ios     ;sector number just read
        cpi     26      ;read last sector?
        jc      rdl
        must be sector 26, zero and go to next track
        lda     iot     ;get track to register a
        inr     a
        mov     c,a     ;ready for call
        call    settrk
        xra     a       ;clear sector number
 il:    inr     a       ;to next sector
        mov     c,a     ;ready for call
        call    setsec
        pop     b       ;recall sector count
        dcr     b       ;done?
        jnz     rdsec

        done with the load, reset default buffer address
ocpm:   ;(enter here from cold start boot)
```

```
        enable rst0 and rst7
        di
        mvi     a,12h   ;initialize command
        out     revrt
        xra     a
        out     intc    ;cleared
        mvi     a,inte  ;rst0 and rst7 bits on
        out     intc
        xra     a
        out     icon    ;interrupt control

        set default buffer address to 80h
        lxi     b,buff
        call    setdma

        reset monitor entry points
        mvi     a,jmp
        sta     0
        lxi     h,wboote
        shld    1       ;jmp wboot at location 00
        sta     5
        lxi     h,bdos
        shld    6       ;jmp bdos at location 5
        if      not test
        sta     7*8     ;jmp to mon80 (may have been changed by ddt)
        lxi     h,mon80
        shld    7*8+1
        endif
;       leave iobyte set
        previously selected disk was b, send parameter to cpm
        lda     cdisk   ;last logged disk number
        mov     c,a     ;send to ccp to log it in
        ei
        jmp     cpmb
;
;       error condition occurred, print message and retry
 oterr:
        pop     b       ;recall counts
        dcr     c
        jz      booter0
        try again
        push    b
        jmp     wboot0

ooter0:
;       otherwise too many retries
        lxi     h,bootmsg
        call    prmsg
        jmp     rmon80  ;mds hardware monitor

 ootmsg:
        db      '?boot',0
;

 onst:  ;console status to reg-a
;       (exactly the same as mds call)
        jmp     csts

onin:   ;console character to reg-a
        call    ci
        ani     7fh     ;remove parity bit
        ret
;
onout:  ;console character from c to console out
        jmp     co
;
```

```
        dad     b       ;translate(sector) address
        mov     a,m     ;translated sector number to A
        sta     ios
        mov     l,a     ;return sector number in L
        ret
;
setdma: ;set dma address given by regs b,c
        mov     l,c
        mov     h,b
        shld    iod
        ret
;
read:   ;read next disk record (assuming disk/trk/sec/dma set)
        mvi     c,readf ;set to read function
        call    setfunc
        call    waitio  ;perform read function
        ret             ;may have error set in reg-a
;
;
write:  ;disk write function
        mvi     c,writf
        call    setfunc ;set to write function
        call    waitio
        ret             ;may have error set

;
;       utility subroutines
prmsg:  ;print message at h,l to 0
        mov     a,m
        ora     a       ;zero?
        rz
;       more to print
        push    h
        mov     c,a
        call    conout
        pop     h
        inx     h
        jmp     prmsg
;
setfunc:
;       set function for next i/o (command in reg-c)
        lxi     h,iof   ;io function address
        mov     a,m     ;get it to accumulator for masking
        ani     11111000b       ;remove previous command
        ora     c       ;set to new command
        mov     m,a     ;replaced in iopb
;       the mds-800 controller requires disk bank bit in sector byte
;       mask the bit from the current i/o function
        ani     00100000b       ;mask the disk select bit
        lxi     h,ios           ;address the sector select byte
        ora     m               ;select proper disk bank
        mov     m,a             ;set disk select bit on/off
        ret
;
waitio:
        mvi     c,retry ;max retries before perm error
rewait:
;       start the i/o function and wait for completion
        call    intype  ;in rtype
        call    inbyte  ;clears the controller
;
        lda     dbank           ;set bank flags
        ora     a               ;zero if drive 0,1 and nz if 2,3
        mvi     a,iopb and 0ffh ;low address for iopb
        mvi     b,iopb shr 8    ;high address for iopb
        jnz     iodr1   ;drive bank 1?
```

```
list:   ;list device out
;       (exactly the same as mds call)
        jmp     lo
;
liststt:
        ;return list status
        xra     a
        ret             ;always not ready
;
punch:  ;punch device out
;       (exactly the same as mds call)
        jmp     po
;
reader: ;reader character in to reg-a
;       (exactly the same as mds call)
        jmp     ri
;
home:   ;move to home position
;       treat as track 00 seek
        mvi     c,0
        jmp     settrk
;
seldsk: ;select disk given by register c
        lxi     h,0000h ;return 0000 if error
        mov     a,c
        cpi     ndisks  ;too large?
        rnc             ;leave HL = 0000
;
        ani     10b     ;00 00 for drive 0,1 and 10 10 for drive 2,3
        sta     dbank   ;to select drive bank
        mov     a,c     ;00, 01, 10, 11
        ani     1b      ;mds has 0,1 at 78, 2,3 at 88
        ora     a       ;result 00?
        jz      setdrive
        mvi     a,00110000b     ;selects drive 1 in bank
setdrive:
        mov     b,a     ;save the function
        lxi     h,iof   ;io function
        mov     a,m
        ani     11001111b       ;mask out disk number
        ora     b       ;mask in new disk number
        mov     m,a     ;save it in iopb
        mov     l,c
        mvi     h,0     ;HL=disk number
        dad     h       ;*2
        dad     h       ;*4
        dad     h       ;*8
        dad     h       ;*16
        lxi     d,dpbase
        dad     d       ;HL=disk header table address
        ret
;
;
settrk: ;set track address given by c
        lxi     h,iot
        mov     m,c
        ret
;
setsec: ;set sector number given by c
        lxi     h,ios
        mov     m,c
        ret
sectran:
        ;translate sector bc using table at de
        mvi     b,0     ;double precision sector number in BC
        xchg            ;translate table address to HL
```

```
        out     ilow            ;low address to controller
        mov     a,b
        out     ihigh   ;high address
        jmp     wait0           ;to wait for complete
;
iodr1:  ;drive bank 1
        out     ilow+10h        ;88 for drive bank 10
        mov     a,b
        out     ihigh+10h
;
wait0:  call    instat          ;wait for completion
        ani     iordy           ;ready?
        jz      wait0
;
;       check io completion ok
        call    intype          ;must be io complete (00) unlinked
;       00 unlinked i/o complete,     01 linked i/o complete (not used)
;       10 disk status changed        11 (not used)
        cpi     10b             ;ready status change?
        jz      wready
;
;       must be 00 in the accumulator
        ora     a
        jnz     werror          ;some other condition, retry

;       check i/o error bits
        call    inbyte
        ral
        jc      wready          ;unit not ready
        rar
        ani     11111110b       ;any other errors?  (deleted data ok)
        jnz     werror
;
;       read or write is ok, accumulator contains zero
        ret
;
wready: ;not ready, treat as error for now
        call    inbyte          ;clear result byte
        jmp     trycount
;
werror: ;return hardware malfunction (crc, track, seek, etc.)
;       the mds controller has returned a bit in each position
;       of the accumulator, corresponding to the conditions:
        0       - deleted data (accepted as ok above)
;       1       - crc error
;       2       - seek error
;       3       - address error (hardware malfumction)
;       4       - data over/under flow (hardware malfunction)
;       5       - write protect (treated as not ready)
;       6       - write error (hardware malfunction)
;       7       - not ready
;       (accumulator bits are numbered 7 6 5 4 3 2 1 0)
;
;       it may be useful to filter out the various conditions,
;       but we will get a permanent error message if it is not
;       recoverable.  in any case, the not ready condition is
;       treated as a separate condition for later improvement
trycount:
;       register c contains retry count, decrement 'til zero
        dcr     c
        jnz     rewait  ;for another try
;
;       cannot recover from error
        mvi     a,1     ;error code
        ret
;
```

```
;       intype, inbyte, instat read drive bank 00 or 10
intype: lda     dbank
        ora     a
        jnz     intyp1  ;skip to bank 10
        in      rtype
        ret
intyp1: in      rtype+10h       ;78 for 0,1  88 for 2,3
        ret
;
inbyte: lda     dbank
        ora     a
        jnz     inbyt1
        in      rbyte
        ret
inbyt1: in      rbyte+10h
        ret
;
instat: lda     dbank
        ora     a
        jnz     instat1
        in      dstat
        ret
instat1:in      dstat+10h
        ret


;
;       data areas (must be in ram)
dbank:  db      0               ;disk bank 00 if drive 0,1
                                ;          10 if drive 2,3
iopb:   ;io parameter block
        db      80h             ;normal i/o operation
iof:    db      readf           ;io function, initial read
ion:    db      1               ;number of sectors to read
iot:    db      offset          ;track number
ios:    db      1               ;sector number
iod:    dw      buff            ;io address
;
;
;       define ram areas for bdos operation
        endef
        end
```

APPENDIX VII

DATA SHEETS OF COMPONENTS

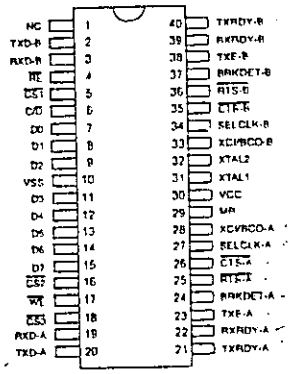
WESTERN DIGITAL
CORPORATION

FINAL

WD2123 DEUCE
Dual Enhanced Universal Communications Element

FEATURES

- TWO INDEPENDENT ASYNCHRONOUS FULL DUPLEX DATA COMMUNICATION CHANNELS (2 BOARTS)
- TWO INDEPENDENT BAUD RATE GENERATORS (ONE PER CHANNEL)
- EACH CHANNEL WITH FOLLOWING FEATURES:
- SELECTABLE 5 TO 8 BIT CHARACTERS
- 1X, 16X, 64X CLOCK RATES
- 16 SELECTABLE BAUD RATE CLOCK FREQUENCIES (INTERNAL)
- LINE BREAK DETECTION AND GENERATION
- 1, 1½, OR 2 STOP BIT SELECTION
- FALSE START BIT DETECTION
- ODD OR EVEN PARITY GENERATE AND DETECTION
- OVERRUN AND FRAMING DETECTION
- DOUBLE BUFFERING OF DATA
- TTL COMPATIBLE INPUTS AND OUTPUTS
- COMPATIBLE WITH 8251A (ASYNC ONLY) AND WD1983 DEVICES
- DIAGNOSTIC LOCAL LOOP-BACK MODE
- RXD INITIALIZATION UPON MASTER RESET
- ON-BOARD OSCILLATOR FOR EASE OF USE WITH A CRYSTAL
- VERSATILE CLOCK SELECT OPTIONS FOR INDEPENDENT TRANSMIT AND RECEIVE RATES



PIN DESIGNATION

DESCRIPTION

The Western Digital WD2123 Dual Enhanced Universal Communications Element (DEUCE) is a single chip MOS/LSI Data Communications Controller Circuit that contains two independent full-duplex asynchronous RECEIVER/TRANSMITTER CHANNELS and two independent BAUD RATE GENERATORS. The WD2123 is fabricated in N-Channel silicon gate technology and is packaged in a 40 pin plastic or ceramic package. All inputs and outputs are TTL compatible.

The WD2123 Block Diagram is shown in Figure 1. The WD2123 is a merger of two WD1983s and one WD1941 from WDC's line of communications devices on one piece of silicon. The 1983 is an asynchronous only version of the 8251A and the 1941 is a baud rate generator. In this manner, 8251A compatibility is maintained with the WD2123 with the added features of 2 channels and 2 baud rate generators on a single chip.

As depicted from the block diagram, the channels are referred to as CHANNELS A and B. CHANNEL A, which is an asynchronous 8251A, is addressed or controlled by the input signal $\overline{CS1}$. CHANNEL B is similarly controlled by $\overline{CS2}$. Finally, the BAUD RATE GENERATORS are controlled by $\overline{CS3}$.

Each channel of the WD2123 can be programmed to receive and transmit asynchronous serial data. The WD2123 performs serial-to-parallel conversion on data characters received from an input/output device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the status of either channel at any time. Status information on a per channel basis reported includes the type and the condition of the transfer operations being performed by the WD2123 as well as any transmission error conditions (parity, overrun, or framing). Programming the WD2123 is identical to the 8251A in the asynchronous mode, remembering that $\overline{CS1}$, when low, selects CHANNEL A and when $\overline{CS2}$ is low, selects CHANNEL B.

The WD2123 BAUD RATE GENERATORS may be selected either internally or externally. The clock select logic includes a clock select control bit CR1 (CS) in each COMMAND INSTRUCTION REGISTER. This control bit allows selection of the internal baud clock or an externally applied clock and works in conjunction with the select clock pin, "SELCLK" and the external clock input/baud clock output pin, "XCI/BCO". When CS is logic 1, the external clock select mode is selected. This means that the transmit and receive clocks (TXC and RXC) are internally tied together and the select clock pin, SELCLK, will determine whether those clocks are driven from the internal baud rate generator (SELCLK is high) or from the external clock input pin, "XCI/BCO", (SELCLK is low).

PIN DESCRIPTION

PIN NUMBER	SIGNAL MNEMONIC	SIGNAL NAME	FUNCTION
10	VSS	GROUND	Ground
30	VCC	POWER SUPPLY	+5VDC power supply input.
7 8 9 11 12 13 14 15	D0 D1 D2 D3 D4 D5 D6 D7	DATA BUS	This is the 8 bit Bidirectional Data Bus. It is the means of communication between the WD2123 and the CPU. Data, control, mode and status registers are accessed via this bus.
5	$\overline{CS1}$	$\overline{CHIP\ SELECT\ ONE}$	V_{IL} on this input selects Channel A and enables computer communications with Channel A Data, control and status registers.
16	$\overline{CS2}$	$\overline{CHIP\ SELECT\ TWO}$	V_{IL} on this input selects Channel B and enables computer communications with Channel B Data, control and status registers.
18	$\overline{CS3}$	$\overline{CHIP\ SELECT\ THREE}$	V_{IL} on this input select the Baud Rate registers for programming.
6	C/$\overline{D}$	CONTROL or $\overline{DATA}$ SELECT	This input is used in conjunction with the appropriate Chip Select and an active read or write operation to determine register access via the Data Bus.
4	$\overline{RE}$	$\overline{READ\ ENABLE}$	V_{IL} on this input allows the CPU to read data, or status information from the selected register.
17	$\overline{WE}$	$\overline{WRITE\ ENABLE}$	V_{IL} on this input allows the CPU to write data or control information into the selected register.
29	MR	MASTER RESET	V_{IH} on this input resets both channels to the idle state and resets the status, command, mode and Data registers.
31	XTAL1	CRYSTAL OSCILLATOR INPUT	This is the input side of the on-chip oscillator. It can also be driven by an external clock source.
32	XTAL2	CRYSTAL OSCILLATOR OUTPUT	This is the output side of the on-chip oscillator.
27	SELCLK-A	SELECT CLOCK (Channel A)	This input is used in conjunction with the Clock Select bit (CR1) in the command register to determine the baud clock source for Channel A.
34	SELCLK-B	SELECT CLOCK (Channel B)	This input is used in conjunction with the Clock Select bit (CR1) in the command register to determine the baud clock source for Channel B.
28	XCI/BCO-A	EXTERNAL CLOCK INPUT/BAUD CLOCK OUTPUT-(Channel A)	This is a bidirectional port, which is used as the externally applied baud clock input or the internal baud rate generator output depending on the states of SELCLK and CR1 command bit. (Channel A)
33	XCI/BCO-B	EXTERNAL CLOCK INPUT/BAUD CLOCK OUTPUT-(Channel B)	This is a bidirectional port, which is used as the externally applied baud clock input or the internal baud rate generator output depending on the states of SELCLK and CR1 command bit. (Channel B)
26	$\overline{CTS-A}$	$\overline{CLEAR\text{-}TO\text{-}SEND}$ (Channel A)	V_{IL} on this input enables Channel A to transmit serial data if the Transmitter is enabled.

PIN NUMBER	SIGNAL MNEMONIC	SIGNAL NAME	FUNCTION
35	$\overline{\text{CTS-B}}$	$\overline{\text{CLEAR-TO-SEND}}$ (Channel B)	V_{IL} on this input enables Channel B to transmit serial data if the Transmitter is enabled.
20	TXD-A	TRANSMIT DATA (Channel A)	This is the Serial Data Output from Channel A.
2	TXD-B	TRANSMIT DATA (Channel B)	This is the Serial Data Output from Channel B.
19	RXD-A	RECEIVE DATA (Channel A)	This is the Serial Data Input for Channel A.
3	RXD-B	RECEIVE DATA (Channel B)	This is the Serial Data Input for Channel B.
21	TXRDY-A	TRANSMITTER READY (Channel A)	This output, when high (V_{OH}), alerts the CPU that Channel A is ready to accept a new data character. The TXRDY output is automatically reset whenever a character is written into the Transmit Holding Register and can be used as an interrupt to the system. $\overline{\text{CTS}}$ must be asserted.
40	TXRDY-B	TRANSMITTER READY (Channel B)	This output, when high (V_{OH}), alerts the CPU that Channel B is ready to accept a new data character. The TXRDY output is automatically reset whenever a character is written into the Transmit Holding Register and can be used as an interrupt to the system. $\overline{\text{CTS}}$ must be asserted.
22	RXRDY-A	RECEIVER READY (Channel A)	This output, when high (V_{OH}), alerts the CPU that Channel B contains a data character that is ready to be input. This output is automatically reset whenever the new character is read from the Receive Holding Register and can be used as an interrupt to the system.
39	RXRDY-B	RECEIVER READY (Channel B)	This output, when high (V_{OH}), alerts the CPU that Channel B contains a data character that is ready to be input. This output is automatically reset whenever the new character is read from the Receive Holding Register and can be used as an interrupt to the system.
23	TXE-A	TRANSMITTER EMPTY (Channel A)	This output, when high (V_{OH}), indicates that Channel A Transmitter has no new characters to send and is waiting in an idle state.
38	TXE-B	TRANSMITTER EMPTY (Channel B)	This output, when high (V_{OH}), indicates that Channel B Transmitter has no new characters to send and is waiting in an idle state.
24	BRKDET-A	BREAK DETECT (Channel A)	This output, when high (V_{OH}), indicates that the Receiver for Channel A has detected a break condition.
37	BRKDET-B	BREAK DETECT (Channel B)	This output, when high (V_{OH}), indicates that the Receiver for Channel B has detected a break condition.
25	$\overline{\text{RTS-A}}$	$\overline{\text{REQUEST-TO-SEND}}$ (Channel A)	A general purpose output that is controlled by the command register bit CR5 for Channel A.
36	$\overline{\text{RTS-B}}$	$\overline{\text{REQUEST-TO-SEND}}$ (Channel B)	A general purpose output that is controlled by the command register bit CR5 for Channel B.
1	NC		No Internal Connection.

If the internal BRG clock is selected, (SELCLK is high) then the external clock input pin becomes a BRG clock output. Hence, the mnemonic, "XCVBCO".

When CR1 (CS) is logic 0, then internal clock select mode is selected. The transmit clock (TXC) is driven by the internal BRG clock and the receive clock is driven by the select clock pin, (SELCLK). The XCVBCO pin becomes the baud clock output (the same signal that is being applied to TXC).

The WD2123 also provides a local loop-back test mode of operation for each channel. This diagnostic mode is independently controlled via the LB(CR7) bit of the COMMAND REGISTER. When LB is logic 1, the channel is programmed for Local Loop-Back. In this diagnostic mode, the TXD output is set to the marking (logic "1") state; the output of the TRANSMIT REGISTER is "looped-back" into the RECEIVER REGISTER input; $\overline{RTS}$ output is held high; the $\overline{CTS}$ and RXD inputs are ignored. An additional requirement is that the TEN(CR0) command bit and the REN(CR2) be logic 1. The status and output flags operate normally.

Each channel is also provided with break character generation and detection. (A break character is defined as all zero data bits, parity bit and stop bits after a valid start bit.) For break character generation, SBRK (CR3) command bit is set to a logic 1. This causes the TXD output to be forced low (spacing) for as long as SBRK is programmed high. The break detect output and status bit (SR6) is set to logic 1, indicating that the receiver has detected a break character. The framing error flag is also set to 1 for this condition.

ARCHITECTURE

The WD2123 is an eight bit bus-oriented device. Communication between the controlling CPU and the two RECEIVER/TRANSMITTER CHANNELS or the two BAUD RATE GENERATORS occurs via the 8-bit data bus through a common set of bus transceivers. Figure 1 is a Block Diagram of the WD2123.

A diagram of one of the two communication controllers is shown in Figure 2. There are two accessible data registers, which buffers transmit and receive data. They are the TRANSMIT HOLDING REGISTER and the RECEIVE HOLDING REGISTER. There is a parallel-to-serial shift register, the TRANSMIT REGISTER and a serial-to-parallel shift register, the RECEIVE REGISTER.

Operational Control and monitoring of the CHANNEL is performed by two CONTROL REGISTERS (the COMMAND INSTRUCTION REGISTER and the MODE INSTRUCTION REGISTER) and the STATUS REGISTER.

A read/write control circuit allows programming/monitoring or loading/reading of data in the CONTROL, STATUS and HOLDING REGISTERS by activating the appropriate control lines: Chip Select ($\overline{CS1}$, $\overline{CS2}$, $\overline{CS3}$), READ ENABLE ($\overline{RE}$), WRITE ENABLE ($\overline{WE}$) and CONTROL or DATA SELECT (C/$\overline{D}$).

Internal control of each channel is by means of two internal microcontrollers: one for transmit and one for receive. The control registers, various counters and external signals provide inputs to the microcontrollers, which generate the necessary control signals to send and receive serial data according to the programmed protocol.

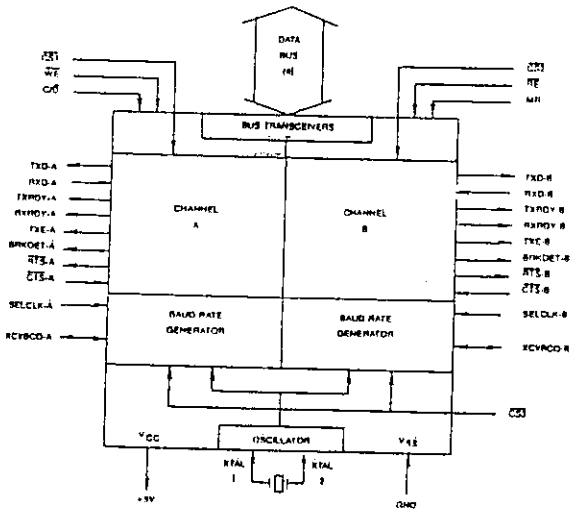


FIGURE 1. WD2123 BLOCK DIAGRAM

A diagram of one of the two BAUD RATE GENERATORS is shown in Figure 3. The 4 low order DATA BUS bits, D0-D3, are used to program the desired rate by loading the RATE REGISTER. Control signals CS3, We and C/D are used to select and load the appropriate register.

The contents of the RATE REGISTER is decoded and addresses a FREQUENCY SELECT ROM for the proper frequency, which is generated by the DIVIDER circuitry and the control logic.

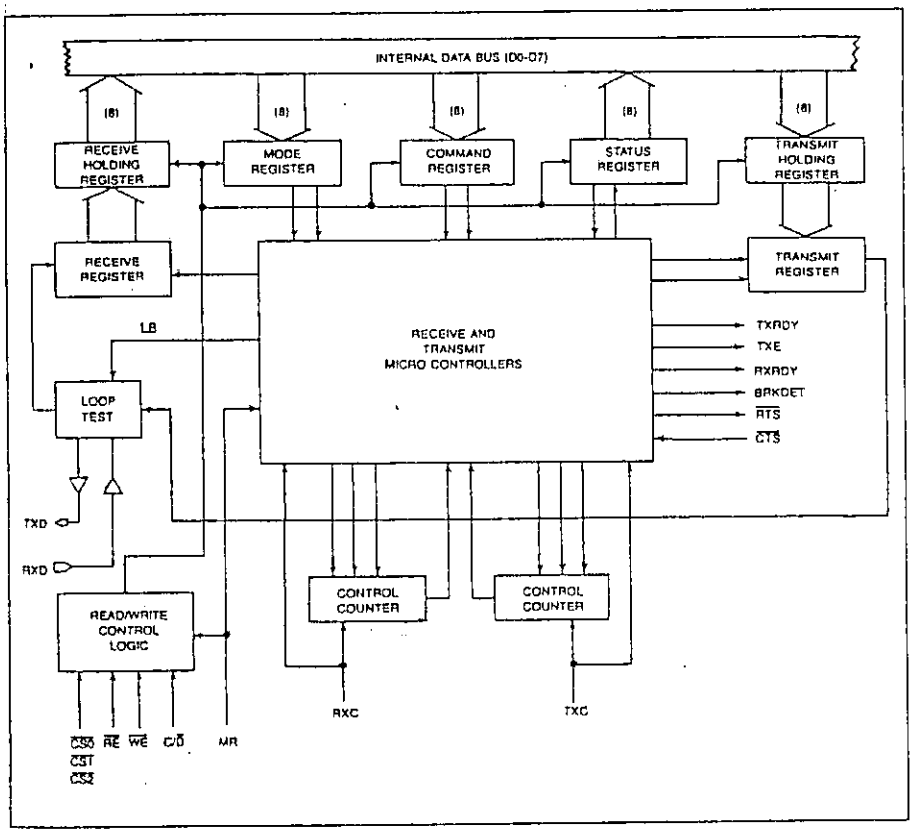


FIGURE 2. RECEIVE/TRANSMIT COMMUNICATIONS CONTROLLER DIAGRAM

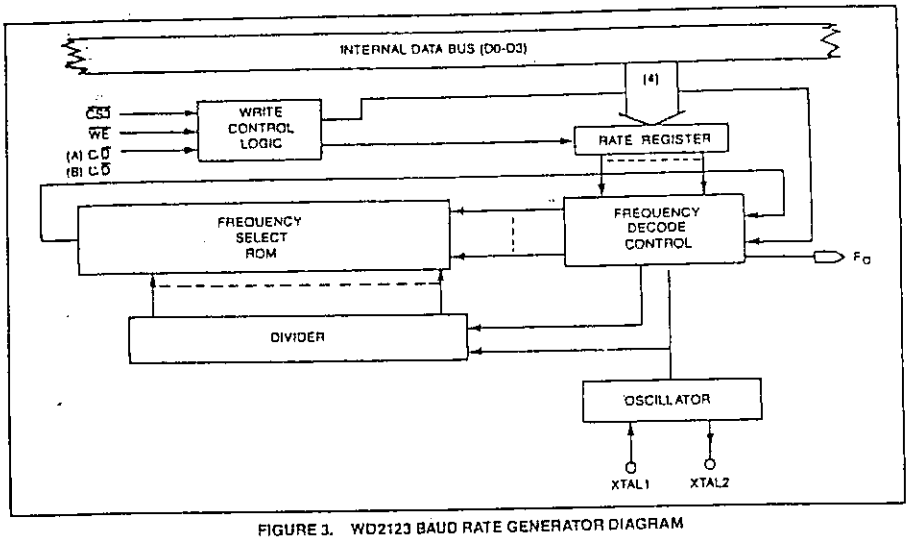


FIGURE 3. WD2123 BAUD RATE GENERATOR DIAGRAM

The WD2123 registers are addressed by the following table:

C/D	RE	WE	CS1	CS2	CS3	REGISTER SELECTED		
L	L	H	L	H	H	RECEIVE HOLDING REG.	—	CHA
L	H	L	L	H	H	TRANSMIT HOLDING REG.	—	CHA
H	L	H	L	H	H	STATUS REG.	—	CHA
H	H	L	L	H	H	MODE AND COMMAND REG.	—	CHA
L	L	H	H	L	H	RECEIVE HOLDING REG.	—	CHB
L	H	L	H	L	H	TRANSMIT HOLDING REG.	—	CHB
H	L	H	H	L	H	STATUS REG.	—	CHB
H	H	L	H	L	H	MODE and COMMAND REG.	—	CHB
L	H	L	H	H	L	RATE REG.	—	CHA
H	H	L	H	H	L	RATE REG.	—	CHB
X	X	X	H	H	H	DATA BUS IN HIGH IMPEDANCE MODE		

TABLE 1. WD2123 REGISTER ADDRESSING

Note:
"L" means V_{IL} at pins.
"H" means V_{IH} at pins.
"X" means don't care.

The WD2123 contains two MODE REGISTERS—one for each channel. The format and definition of the MODE REGISTERS are shown below:

MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
S2	S1	EP	PEN	L2	L1	B2	B1

B2	B1	BAUD RATE FACTOR
0	0	Undefined
0	1	1X
1	0	16X
1	1	64X
L2	**L1**	**CHARACTER LENGTH**
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits
PEN		**PARITY ENABLE**
0		Disable Parity
1		Enable Parity
EP		**PARITY SELECT**
0		Odd Parity
1		Even Parity
S2	**S1**	**NUMBER OF STOP BITS**
0	0	Invalid
0	1	1 Bit
1	0	1½ Bits *
1	1	2 Bits

TABLE 2. WD2123 MODE REGISTERS

* 16X and 64X only. 1X will be 2 stop bits.

The WD2123 contains two COMMAND REGISTERS—one per channel. The format and definition of the COMMAND REGISTERS are shown below:

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
LB	IR	RTS	ER	SBK	REN	CS	TEN

TEN	TRANSMIT ENABLE
1	Enable
0	Disable
CS	**CLOCK SELECT**
1	XMIT and RCV Clock source common
0	XMIT and RCV Clock sources different
REN	**RECEIVE ENABLE**
1	Enable
0	Disable
SBK	**SEND BREAK CHARACTER**
1	Force TXD Low
0	Normal Operation
ER	**ERROR RESET**
1	Reset Error Flags
0	No Reset
RTS	**REQUEST TO SEND**
1	Force $\overline{RTS}$ pin = 0 (V_{OL})
0	Force $\overline{RTS}$ pin = 1 (V_{OH})
IR	**INTERNAL RESET**
1	Next Write to Mode Register
0	Next Write to Command Register
LB	**LOOP BACK ENABLE**
0	Normal Operation Mode
1	Local Loop-Back Mode

TABLE 3. WD2123 CONTROL REGISTERS

The WD2123 contains two STATUS REGISTERS—one per channel. The STATUS REGISTER is a read-only register. The format and definition of the STATUS REGISTERS are shown below:

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
CTS	BRK DET	FE	OE	PE	TXE	RX RDY	TX RDY

TXRDY	**TRANSMITTER READY**
1	Denotes THR is empty and ready for a new character
0	THR not empty. (Reset when THR is loaded by CPU)
RXRDY	**RECEIVER READY**
1	Denotes that the RHR contains a valid character
0	RHR does not contain a valid character. (Reset when the CPU reads the RHR)
TXE	**TRANSMITTER EMPTY**
1	Denotes that the TR is empty
0	Denotes that the TR is not empty
PE	**PARITY ERROR**
1	Denotes Parity Error
0	No Parity Error. (Reset by ER bit of command register)
OE	**OVERRUN ERROR**
1	Denotes Overrun Error
0	No Overrun Error. (Reset by ER bit of command register)
FE	**FRAMING ERROR**
1	Denotes Framing Error
0	No Framing Error. (Reset by ER bit of command register)
BRKDET	**BREAK DETECT**
1	Indicates that the receiver has detected a line break condition. (FE will also be set)
0	No Break Condition detected for at least one bit time
CTS	**CLEAR-TO-SEND**
1	Indicates that the $\overline{CTS}$ pin is active (V_{IL})
0	Indicates that the $\overline{CTS}$ pin is not active (V_{IH})

TABLE 4. WD2123 STATUS REGISTERS

The WD2123 contains two RATE REGISTERS that are used to select 16 BAUD rates when CR1 = 1 and SELCLK = 1. The Format of the RATE REGISTERS is shown below. Note that the Receiver and the Transmitter of any channel run off the same Baud clock except when CR1 = 0, then the Transmitter runs off the Baud Clock and the Receiver runs off an externally applied signal input on the SELCLK pin.

D7							D0
X	X	X	X	RA3 RB3	RA2 RB2	RA1 RB1	RA0 RB0

When C/D = 0, RA3 to RA0 are loaded.
When C/$\overline{D}$ = 1, RB3 to RB0 are loaded.

The C/$\overline{D}$ line is used in conjunction with $\overline{CS3}$ and $\overline{WE}$ to program the desired BAUD rate. When C/$\overline{D}$ is low, Channel A is selected, and when C/$\overline{D}$ is high, Channel B is selected. The low order 4 bits of the DATA BUS are loaded into the selected rate register, and the high order 4 bits are ignored.

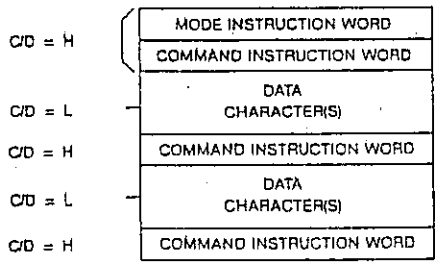
When the crystal frequency equals 1.8432 MH_z the following baud rates may be programmed.

				BAUD RATE			FREQUENCY	
R3	R2	R1	R0	BRF 1X	BRF 16X	BRF 64X	(KHZ)	DIVISOR
0	0	0	0	800	50.0	12.50	0.80	2304
0	0	0	1	1,200	75.0	18.75	1.20	1536
0	0	1	0	1,760	110.0	27.50	1.76	1049
0	0	1	1	2,150	134.5	33.59	2.15	855
0	1	0	0	2,400	150.0	37.50	2.40	768
0	1	0	1	3,200	200.0	50.00	3.20	576
0	1	1	0	4,800	300.0	75.00	4.80	384
0	1	1	1	9,600	600.0	150.00	9.60	192
1	0	0	0	19,200	1,200.0	300.00	19.20	96
1	0	0	1	28,800	1,800.0	450.00	28.80	64
1	0	1	0	38,400	2,400.0	600.00	38.40	48
1	0	1	1	57,600	3,600.0	900.00	57.60	32
1	1	0	0	76,800	4,800.0	1,200.00	76.80	24
1	1	0	1	115,200	7,200.0	1,800.00	115.20	16
1	1	1	0	153,600	9,600.0	2,400.00	153.60	12
1	1	1	1	307,200	19,200.0	4,800.00	307.20	6

TABLE 5. WD2123 BAUD RATE SELECTION

READ/WRITE OPERATIONS

The WD2123 must be initialized after a MASTER RESET pulse by first writing the MODE INSTRUCTION word and then the COMMAND INSTRUCTION word. Thereafter, every control write to the device is interpreted as a COMMAND word. If it is desired to re-program the MODE REGISTER, a COMMAND REGISTER bit, INTERNAL RESET (CR6), allows the next control write data to be entered into the MODE REGISTER.



TYPICAL DATA BLOCK TRANSFER

OPERATING DESCRIPTION

The WD2123 is primarily designed to operate in an 8 bit microprocessor environment, although other control logic schemes are easily implemented. The DATA BUS and the interface control signals ($\overline{CS1}$, $\overline{CS2}$, $\overline{CS3}$, C/$\overline{D}$, $\overline{RE}$, $\overline{WE}$) should be connected to the microprocessor's data bus and system control bus. A 1.8432 MHz crystal should be connected to the WD2123 as shown in figure 5. The appropriate TXC (RXC) clock frequencies should be programmed via system software. Different Baud clock configurations are possible, such as separate transmit and receive frequencies, and are outlined in the general description.

For typical data communication applications, the RXD and TXD input/outputs can be connected to RS-232C interface circuits. Interface control signals, $\overline{CTS}$ and $\overline{RTS}$, are controlled and sensed by the CPU through the COMMAND and STATUS REGISTERS and can be configured in several ways. The $\overline{CTS}$ input can be used to synchronize the transmitter to external events.

The TXRDY, RXRDY, TXE and BRKDET FLAGS may be connected to the microprocessor system as interrupt inputs or the STATUS REGISTER can be periodically read in a polled environment to support data communication control operations.

The SBRK bit of the COMMAND REGISTER (CR3) is used to send a Break Character. (A Break Character is defined as a start bit, and all zero data, parity and stop bits.) When the CR3 bit is set to a "1", it causes the transmitter output, TXD, to be forced low after the last bit of the last character is transmitted.

The Receiver is equipped with logic to look for a break character. When a break is received, the BREAK DETECT (BRKDET) FLAG and STATUS bit are set to "1". When the receiver input line goes high (V_{IH}) for at least one clock period, the receiver resets the BRKDET FLAG and resumes its search for a start bit.

PROGRAMMING PROCEDURE

The programming sequence of the two channels will be different, depending on whether it is an initialization sequence (that is, one performed right after a hardware master reset occurs) or a re-programming sequence (that is, one performed to change the protocol characteristics (Parity, rate, character length, etc.) after the device has been previously operating in the system). The programming sequence differs, in that, after a master reset, the chip is set to expect the first *control write* operation (C/$\overline{D}$ = 1) to contain a *mode instruction*. Any *subsequent control write* operations will be transferred to the *command instruction* register.

Now when it is desired to *change* the *mode instruction* register contents, the following re-programming sequence should be performed. A Command Control word of "40" Hex is written to the Chip. This turns off the Receiver and Transmitter and sets the IR (Internal Reset) bit. This bit causes the read/write control logic to expect the *next control write* operation to be a *new mode instruction*. After the new mode instruction is written to the chip, all subsequent control write operations will again be interpreted as *command instructions*. Therefore, after the *new mode instruction* is performed, the next command would turn the receiver and transmitter back on and resume normal Data operations.

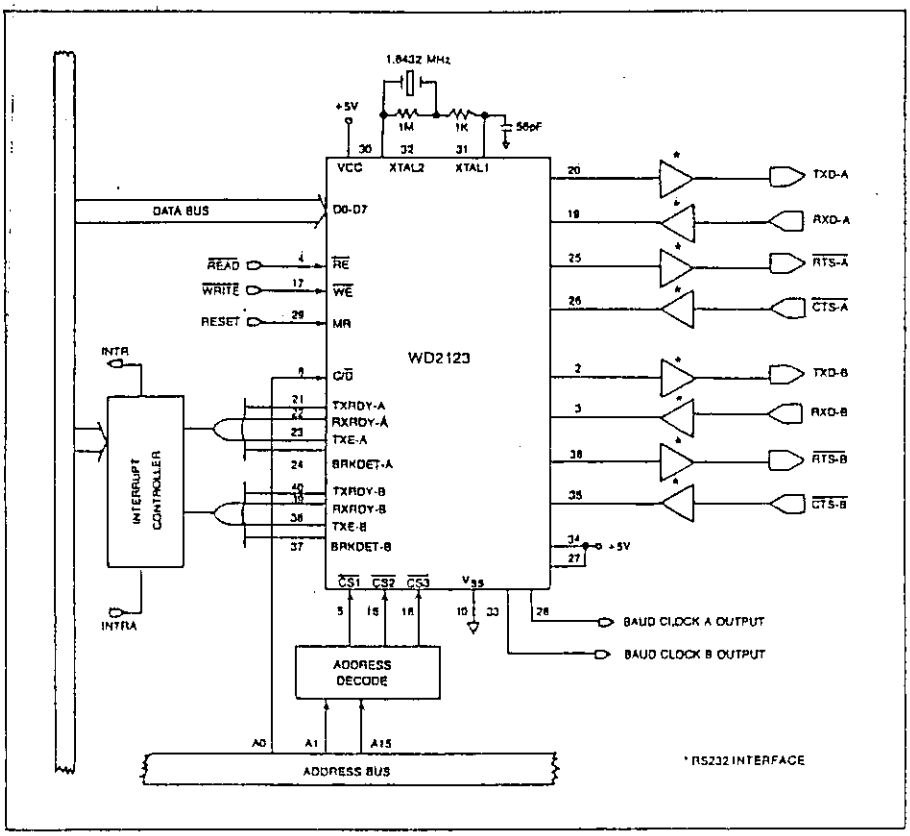


FIGURE 4. WD2123 MICROPROCESSOR APPLICATION

ABSOLUTE MAXIMUM RATINGS

V_{DD} with respect to V_{SS}	0.5V to + 12V
Voltage on Any Pin with Respect to Ground	−0.5V to +7V
Power Dissipation	500 Mw.
Lead Temperature (Soldering 10 sec.)	300°C

STORAGE TEMPERATURE:
Ceramic: —65°C to +150°C
Plastic: —55°C to +125°C

CRYSTAL SPECIFICATIONS:

Temperature range	0°C to +70°C
Series resistance	300Ω to 500Ω
Overall tolerance	±0.01%

Note: Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under dc electrical characteristics.

TABLE 6. DC ELECTRICAL CHARACTERISTICS

T_A = 0°C to +70°C; V_{CC} = 5.0V ±5%; GND = 0V

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	TEST CONDITIONS
V_{IL}	Input Low Voltage	−0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −100 uA
I_{DL}	Data Bus Leakage (High Impedance State)			−50 10	uA uA	V_{OUT} = 0.45V V_{OUT} = V_{CC}
I_{IL}	Input Leakage			10	uA	V_{IN} = V_{CC}
I_{CC}	Power Supply Current		100	125	mA	V_{CC} = 5.25V No Load

TABLE 7. CAPACITANCE

T_A = 25°C; V_{CC} = GND = 0V

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	TEST CONDITIONS
C_{IN}	Input Capacitance			10	pF	f_C = 1MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND.

AC ELECTRICAL CHARACTERISTICS

TABLE 8. A.C. CHARACTERISTICS

T_A = 0°C to + 70C; V_{CC} = 5.0V ±5%; GND = 0V

SYMBOL	CHARACTERISTIC	MIN	MAX	UNITS	CONDITIONS
BUS PARAMETERS Read Cycle					
t_{AR}	Address Stable Before READ ($\overline{CS}$,C/$\overline{D}$)	50		ns	
t_{RA}	Address Hold Time for READ ($\overline{CS}$,C/$\overline{D}$)	50		ns	
t_{RE}	READ Pulse Width	230		ns	
t_{RD}	Data Delay from READ		200	ns	C_L = 50 pF
t_{RDH}	READ to Data Floating	25	200	ns	C_L (Max) = 50 pF C_L (Min) = 15 pF
Write Cycle					
t_{AW}	Address Stable Before WRITE	50		ns	
t_{WA}	Address Hold Time for WRITE	50		ns	
t_{WE}	WRITE Pulse Width	230		ns	
t_{DS}	Data Set-Up Time for WRITE	TWE		ns	
t_{WDH}	Data Hold Time for WRITE	100		ns	
OTHER TIMINGS					
t_{TXC}	Transmit Clock Period	1.6		us	
t_{DTX}	TxD Delay from Falling Edge of TxC		1000	ns	C_L = 100 pF
t_{SRX}	Rx Data Set-Up Time to Sampling Pulse	200		ns	C_L = 100 pF
t_{HRX}	Rx Data Hold Time to Sampling Pulse	200		ns	C_L = 100 pF
t_{TX}	Transmitter Input Clock Frequency 1x Baud Rate 16x and 64x Baud Rate	 DC DC	 500 600	 kHz kHz	Clock 50% Duty Cycle
t_{TPW}	Transmitter Input Clock Pulse Width 1x Baud Rate 16x and 64x Baud Rate	 1.0 800		 us ns	
t_{TPD}	Transmitter Input Clock Pulse Delay 1x Baud Rate 16x and 64x Baud Rate	 1.0 800		 us ns	

TABLE 9. A.C. CHARACTERISTICS (CONTINUED)

SYMBOL	CHARACTERISTICS	MIN	MAX	UNIT	TEST CONDITION
t_{RX}	Receiver Input Clock Frequency 1x Baud Rate 16x and 64x Baud Rate	 DC DC	 500 600	 kHz kHz	Clock 50% Duty Cycle
t_{RPW}	Receiver Input Clock Pulse Width 1x Baud Rate 16x and 64x Baud Rate	 1.0 800		 us ns	
t_{RPD}	Receiver Input Clock Pulse Delay 1x Baud Rate 16x and 64x Baud Rate	 1.0 800		 us ns	
t_{TX}	TxRDY Delay from Center of Stop Bit		8	t_{RXC}	C_L = 50pF (16X)
t_{RX}	RxRDY Delay from Center of Stop Bit		1	t_{RXC}	
t_{IS}	Internal BRKDET Delay from Center of Data Bit		1	RXC	
t_{TRD}	TxRDY Delay from Falling Edge of WRITE		450	ns	
t_{TOD}	TXD Output from Falling Edge of WRITE		1½	t_{TXC}	
t_{WC}	Control Delay from Rising Edge of WRITE ($\overline{RTS}$)		200	ns	
t_{CR}	Control to READ Set-Up Time ($\overline{CTS}$)		1	t_{TXC}	
t_{MR}	Master Reset	500		ns	

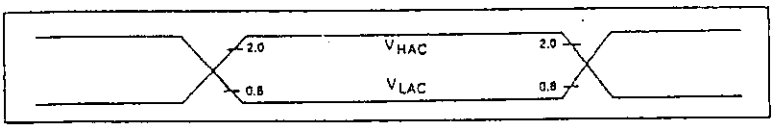


FIGURE 5. A.C. TEST POINTS

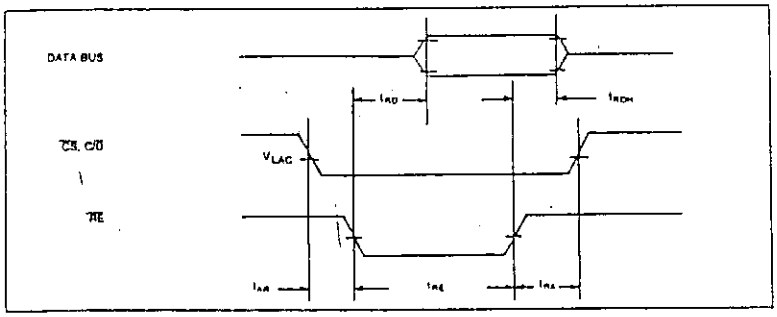


FIGURE 6. READ TIMING

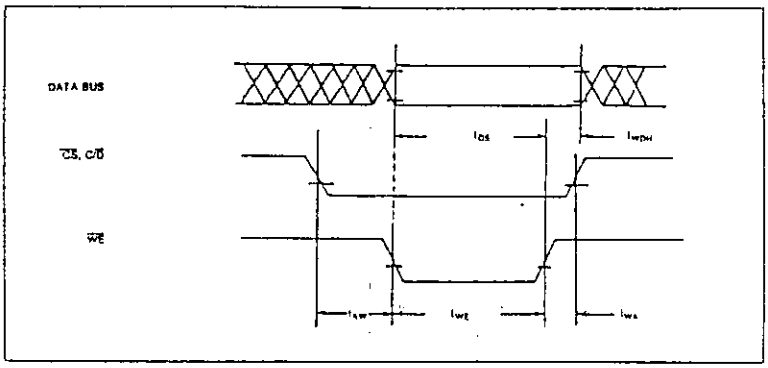


FIGURE 7. WRITE TIMING

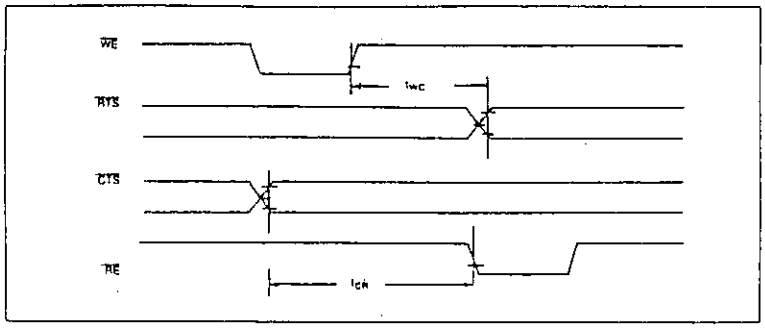


FIGURE 8. INTERFACE CONTROL TIMING

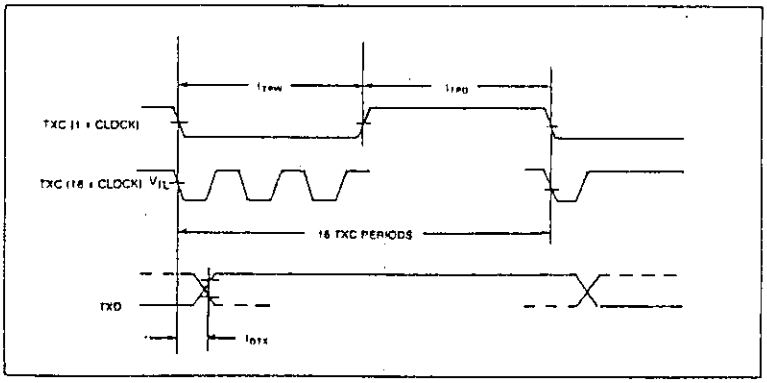


FIGURE 9. TRANSMITTER CLOCK AND DATA TIMING

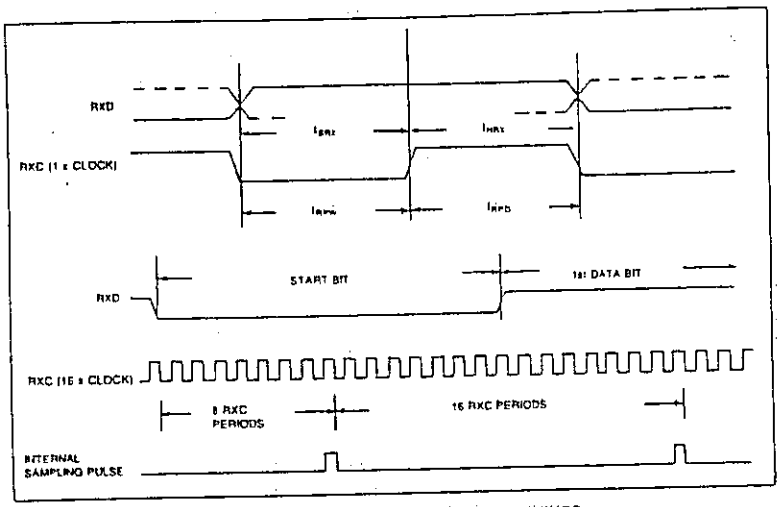


FIGURE 10. RECEIVER CLOCK AND DATA TIMINGS

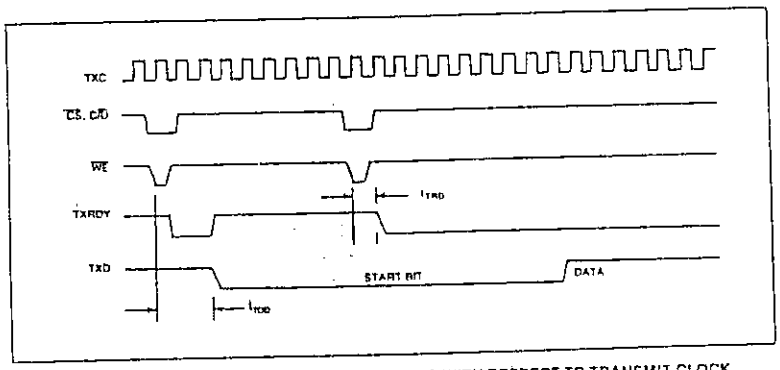


FIGURE 11. TRANSMITTER OUTPUT TIMINGS WITH RESPECT TO TRANSMIT CLOCK

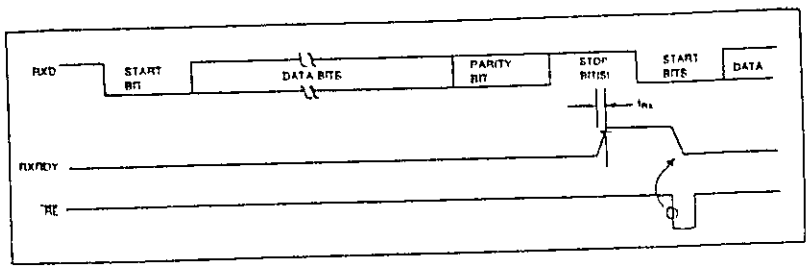


FIGURE 12. RXRDY TIMING

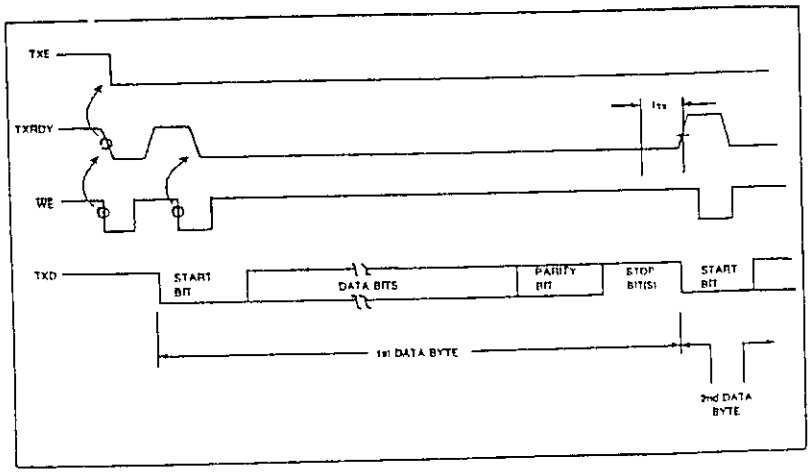
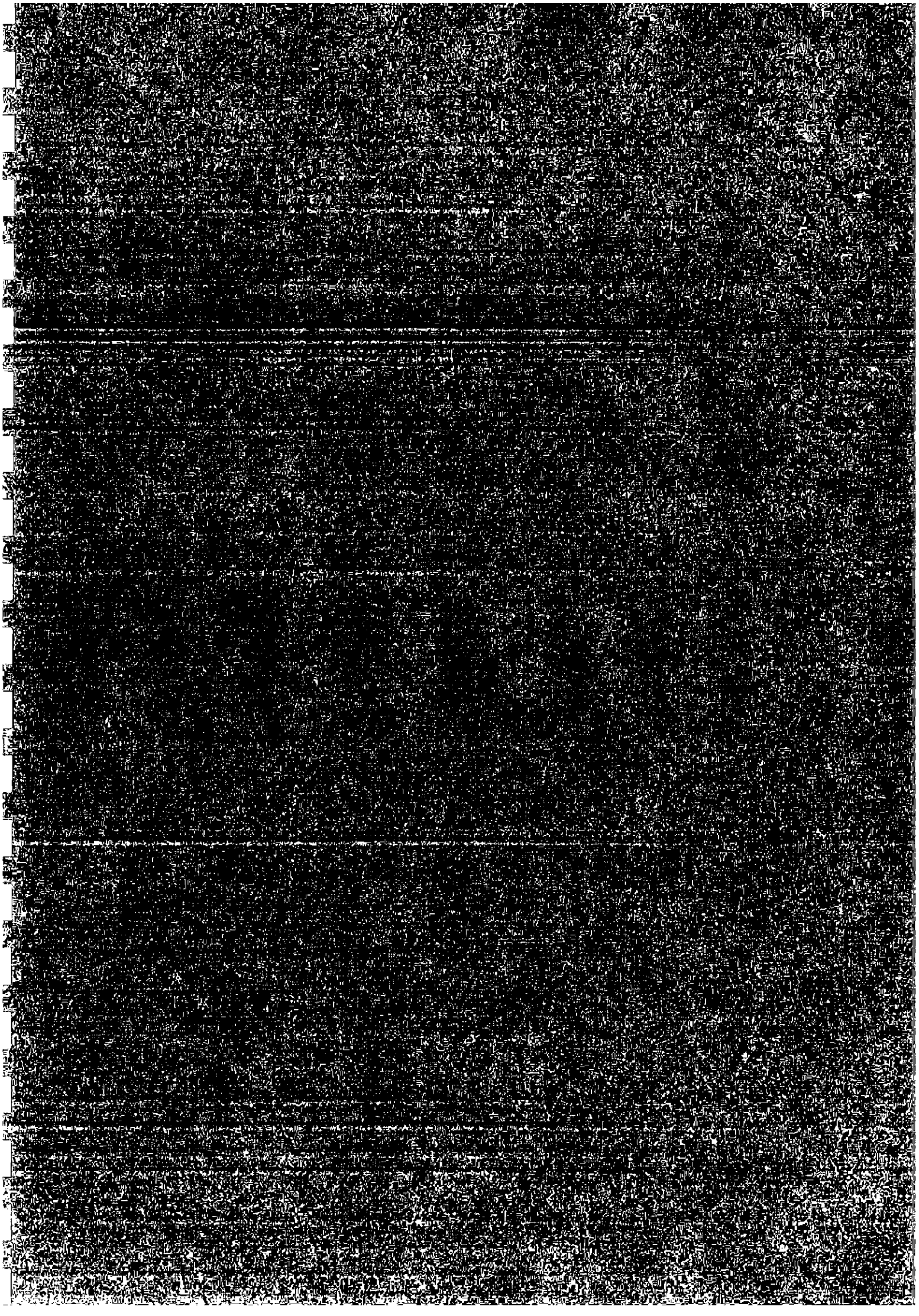


FIGURE 13. TXRDY TIMING



AKNALUU

- .Hardware.

O hardware do sistema de diskettes TIMEX, daqui para a frente referido apenas por FDD, e constituido por duas partes basicas: o conjunto formado pelo controlador, as unidades de disco e a fonte de alimentacao e o "interface" que estabelece a ligacao entre esta e o Spectrum e inclui a extensao do BASIC do Spectrum (FDD-BASIC).

.a - Interface.

O interface como se pode ver no esquema anexo, dispoe de 4Kb de ROM ou EPROM, onde residem as rotinas de extensao do BASIC e comunicacao com a unidade de discos, 1Kb de RAM, utilizada para armazenamento de variaveis de sistema e buffers de dados, para as rotinas de comunicacao, um porto de comunicacoes bidirecional e logica de comando.

O mapa de memoria do interface e o seguinte:

0000h - 1FFFh	ROM
2000h - 3FFFh	RAM
0EFh	PORTO de COMUNICACOES

O espaco de enderecos ocupado, nao corresponde ao tamanho das memorias utilizadas, existindo portanto varios ecos para cada memoria.

Como se pode observar, o espaco usado e o mesmo da ROM do Spectrum. O conflito e resolvido pela logica do interface que selecciona uma ou outra ROM, conforme o estado de um biestavel interno. Este mecanismo e actuado pelo processador, ao executar um ciclo de "operation code fetch", em determinados enderecos. Assim 0000h e 0008h selecionam a ROM de extensao e 0004h a ROM do Spectrum.

A escolha dos enderecos de entrada na ROM de extensao, foi feita de modo a ganhar o controle do computador, apos um "cold start" (0000h) e sempre que o Spectrum detecte um erro de funcionamento (0008h). A reentrada no Spectrum e feita por um salto indirecto, atraves da instrucao RET que se encontra na posicao 0004h da nova ROM.

1.b - Unidade de discos.

1.b.1 - Controlador.

Esta unidade e relativamente complexa pelo que a descricao sera sumaria.

O controlador, e um computador autonomo baseado no microprocessador Z80, com 16 Kb de RAM, 1 Kb de ROM, controlador de discos, controlador de canais asincronos, porto bidireccional de comunicacoes e logica de comando.

A ROM, que esta activa logo apos um reset, tem um pequeno programa que le para memoria, a partir de 3F00h, o conteudo do sector 0, da pista 0, do disco A e termina com uma instrucao JP 3F00h.

Fazendo o set do bit 6 do porto 0E0h do controlador, esta ROM e desactivada e a RAM passa a ocupar os primeiros 16 Kb de espaco de

memoria. E este o processo utilizado para carregar o TOS em memoria, apos um reset. No primeiro sector das diskettes distribuidas com o sistema, esta um programa que executado a partir de 3F00h instala o TOS.

O controlador de discos e um circuito integrado 1770, da firma Western Digital, da mesma familia que a serie 179X e 279X da mesma marca. Do ponto de vista da programacao, e compativel com estes, tendo como unica limitacao, o facto de nao suportar unidades de disco de 8 polegadas. A base de enderecamento do integrado e o porto 800h. O hardware suporta ate 4 discos simultaneamente. O cabo de ligacao e standard para drives de 5 1/4 polegadas, com as quais as drives do FDD sao compativeis.

O controlador asincrono e um circuito integrado 2123, da firma Western Digital e inclui dois canais asincronos, totalmente independentes que estao ligados, por intermedio de conversores de nivel, as fichas marcadas canal_a e canal_b. Este circuito inclui ainda, um "baud rate generator" duplo que comanda a velocidade de funcionamento dos dois canais. E possivel programar este circuito para diversas combinacoes de velocidade, paridade, numero de bits por caracter, etc.. As bases de enderecamento para este circuito sao 88h e 40h, respectivamente para o canal_a e canal_b e 10h para o "baud rate generator".

Para informacao mais detalhada sobre estes dois circuitos integrados, consultar a documentacao da Western Digital em anexo.

Existem no controlador mais dois portos de I/O para comando do sistema. Um e o porto de comunicacoes com o Spectrum, onde o bit 7 e usado para leitura do sinal DRQ do controlador de discos e o outro suporta os seguintes sinais.

bit 0 - Drive select 0 - Escolha da unidade de disco activa
bit 1 - Drive select 1 - Idem
bit 2 - Drive select 2 - Idem
bit 3 - Drive select 3 - Idem
bit 4 - Side select - Lado da diskette a usar
bit 5 - Double Density - Seleccao da densidade do disco
bit 6 - Boot - Paginacao da ROM de Bootstrap
bit 7 - In Use - Luz dos drives

Estes dois portos estao respectivamente em 2Fh e 9E0h.

1.b.2 - Unidades de disco.

As unidades de disco fornecidas com o FDD, suportam densidade simples e dupla, tem 40 pistas e um tempo de acesso pista a pista de 3 mS. A capacidade total nao formatada e de 250Kb.

Dentro em breve, havera a disposicao do publico unidades de disco com 80 pistas e dupla face, tendo portanto, uma capacidade total nao formatada de 1Mb.

Estas unidades usam o BUS standard SHUGART, para discos de 5" 1/4, de modo que o sistema FDD podera funcionar com outras unidades de disco de caracteristicas semelhantes.

1.b.3 - Fonte de Alimentacao.

A fonte gera +5V e +12V e pode suportar um controlador e duas unidades de disco.

2 - Software.

O programa FDD-BASIC contido na EPROM do interface e composto por varios modulos, dos quais destacamos a analise sintatica, as rotinas de comunicacao e as rotinas de execucao.

2.a - Analise sintatica.

A extensao do BASIC do Spectrum e baseada no prolongamento da analise sintatica das linhas de programa levando em conta os novos comandos.

Todos os erros detectados pelo Spectrum dao origem a um salto para o endereco 0008h, onde tem inicio a rotina de processamento de erros. A passagem do programa por este local activa o mecanismo de paginacao e o programa de extensao fica em controle do computador.

Quando e detectado um erro, numa linha de programa ou comando directo, o programa de extensao verifica se o "erro" foi provocado por um dos novos comandos, analisando a linha em causa a partir do local onde o Spectrum detectou o erro. Caso esta analise falhe, o estado do computador e reposto e o comando devolvido ao Spectrum, exactamente na mesma posicao onde tinha sido retirado, prosseguindo o processamento normal de erro.

A sintaxe do novo comando e verificada e, quando o Spectrum se encontra em modo de execucao, o comando entregue a rotina respectiva. Quando esta termina e chamada uma rotina de processamento de fim de comando que pode, em alguns casos , entregar a iniciativa ao TOS (Ex: escrita no ecran no caso de CAT*) e que termina processando os eventuais erros e devolvendo o comando ao Spectrum.

Caso o Spectrum esteja apenas em modo de analise sintatica, por exemplo quando se esta a criar um programa, o sinal de erro e eliminado e o controle devolvido ao Spectrum que aceita a linha incluindo o novo comando.

Quando a analise da nova sintaxe falha, o controle e devolvido ao Spectrum, por intermedio de um salto para o endereco guardado na variavel de sistema VECTOR. Apos um reset esta variavel e inicalizada com o endereco da rotina de retorno ao Spectrum, mas pode ser alterada para qualquer outro valor, dando a possibilidade ao utilizador de criar os seus proprios comandos com a sintaxe que desejar.

No fim dos novos comandos, o retorno ao Spectrum e feito por um salto para o endereco que se encontra na variavel ABORT, o que permite ao utilizador interceptar o retorno ao Spectrum e alterar o modo como os comandos terminam. Esta facilidade e util para alterar o processamento dos erros.

No capitulo de exemplos sao desenvolvidas aplicacoes destas possibilidades.

2.b -Comunicacoes.

A troca de informacao entre o Spectrum e a unidade de disco e feita por um conjunto de rotinas que enviam num sentido e noutro pacotes de informacao.

Cada pacote e composto por um cabecalho, os dados propriamente ditos e um byte que e a soma truncada de todos os bytes anteriores (checksum). So sao considerados dois tipos de pacotes: dados e comandos.

Nas comunicacoes admite-se que o Spectrum tem sempre a prioridade, so a cedendo quando o indica expressamente. Nao ha deste modo conflitos em relacao a utilizacao do canal de comunicacoes.

Todos os comandos sao executados pelo Spectrum de um modo semelhante. Caso haja dados estes sao enviados primeiro e o sistema de discos guarda-os num buffer interno, em seguida e enviado o comando,

ficando o Spectrum a aguardar a resposta que inclui uma mensagem de erro caso seja o caso. Para exemplificar vamos simular do ponto de vista de estrutura a abertura de um ficheiro (um exemplo detalhado sera dado mais tarde).

1 - Enviar o nome do ficheiro a abrir.
2 - Enviar o comando de abertura com os parametros necessarios para identificar o modo de abertura.
3 - Receber a indicacao de fim de comando.
4 - Processar o eventual erro (Ex: ficheiro nao existe).

Para facilitar estas tarefas existem varias rotinas que estao descritas no bloco seguinte.

2.c - Tabela de Saltos.

Tendo em vista o programador em assembler existe a partir do endereco 0605h uma tabela de saltos que da acesso as principais rotinas da ROM de extensao. Esta tabela sera mantida constante ao longo das diversas versoes do programa que possam vir a ser lancadas, de modo a manter compatibilidade com todos os programas em assembler desenvolvidos pelos utilizadores.

O uso de rotinas que nao constem da tabela ou directamente dos enderecos das que constam pode levar a problemas de incompatibilidade e nao e recomendado.

No ultimo capitulo ha uma lista de todas elas com uma explicacao detalhada do seu funcionamento e dos seus parametros. Pretende-se aqui dar uma visao geral das mesmas.

Estas rotinas dividem-se em 3 grupos:

1 - Comunicacoes - Rotinas de transmissao e recepcao.
2 - Tranferencia de conteudo de memoria - Variantes de SAVE e LOAD.
3 - Utilitarios.

No primeiro grupo encontram-se as seguintes:

PUTDAT - Envia bloco de dados para o FDD.
PUTCOM - Envia comando para o FDD.
GETBLOCK - Recebe dados ou comandos do FDD.
SENDBL - Envia pacote para o FDD (uso especializado).
GETBL - Recebe pacote do FDD.

No segundo:

SAVEP - Guarda em disco programa ou memoria.
LOADP - Carrega do disco programa ou memoria.
WRTMEM - Guarda codigo num ficheiro.
RDMEM - Le codigo de um ficheiro.
RDBLOC - Le ate 256 bytes de um ficheiro.

No ultimo:

CBMS - Executa rotinas do Spectrum chamando-as da extensao.
RESPOSTA - Terminacao de comandos.

O processo de comunicacao e delicado pelo que se recomenda que as rotinas de comunicacao sejam usadas sem alteracao.

As rotinas SENDBL e GETBL nao testam a condicao de "BREAK" e tentam a comunicacao apenas dez vezes, pelo que devem ser usadas quando o utilizador incluir o seu proprio processamento de erro, ou pretender um maior controle sobre o modo como as comunicacoes se processam. Elas sao a base de todas as outras rotinas de comunicacao e podem ser usadas para

mplementar variantes das outras rotinas disponiveis. Na realidade, as restantes rotinas disponiveis na tabela, nao sao essenciais e sao apresentadas apenas para evitar duplicacao de codigo pela parte do tilizador.

Todas estas rotinas, variaveis e buffers estao na pagina de extensao de BASIC e, para ter acesso a elas e necessario selecionar-la, o que se consegue actuando o mecanismo de paginacao. Chama-se a atencao para o facto do mecanismo de paginacao afectar apenas o espaco de memoria de 0000h a 3F00h estando a RAM sempre acessivel.

Para selecionar a pagina de extensao e necessario fazer um CALL ao endereco 0008h com 0 no registo IY. Se este registo nao for 0 o programa de extensao assume que se trata de um erro convencional do Spectrum e tenta processa-lo, o que possivelmente sera desastroso. A rotina sugerida para actuar o mecanismo e a seguinte:

```
PUSH    IY              ;preservar IY
LD      IY,0            ;valor requerido
CALL    8               ;o controle e devolvido
POP     IY              ;recuperar IY
```

Ao seleccionar a pagina de extensao e executada a instrucao DI pelo que os sinais de interrupcao deixam de ser atendidos. Existe no entanto uma instrucao RET no endereco 0038h para que no caso do utilizador os ligar com o processador em modo 1, os efeitos nao sejam desastrosos. No endereco 0603h esta uma instrucao EI de modo a que ao reentrar na ROM do Spectrum as interrupcoes voltem a ser atendidas. O utilizador pode portanto optar por um CALL 0603h ou CALL 0604h para selecionar a ROM do Spectrum respectivamente com e sem ligar o processamento de "interrupts". Faz-se notar que enquanto a ROM de extensao estiver selecionada as rotinas de teclado nao sao executadas e portanto as variaveis associadas KSTATE, LAST_K e FRAMES nao sao actualizadas.

O processamento de NMI, tal como acontece no Spectrum, nao esta disponivel ao utilizador.

Para obter informacao mais detalhada sobre o conteudo da ROM FDD-BASIC recomenda-se a seguinte rotina:

```
        PUSH    IY              ;paginacao
        LD      IY,0
        CALL    8
        POP     IY
;
        LD      HL,0            ;inicio da ROM
        LD      DE,BUFFER       ;local disponivel em RAM
        LD      BC,4096         ;tamanho da ROM
        LDIR                    ;tranferencia do codigo
;
        JP      603H            ;retorno ao Spectrum ligando
                                ;interrupts
```

Esta rotina transfere o conteudo da ROM FDD-BASIC para RAM onde pode ser analisada por um programa monitor.

- Exemplos.

Neste capitulo, vao ser desenvolvidos alguns exemplos, com o propocito de orientar o programador em assembler.

Assume-se que: o utilizador e razoavelmente fluente em programacao assembler do Z80; conhece as rotinas em ROM e o funcionamento do Spectrum dispoe dos meios necessarios para o desenvolvimento de programas naquela linguagem. As menemonicas usadas nos exemplos sao Zilog standard.

O uso de programas monitor para o teste das rotinas desenvolvidas e delicado. Devido a paginacao, nao e possivel usar as possibilidades de trace" e "breakpoint" nos trocos de codigo que sao executados recorrendo a ROM FDD-BASIC, nem e possivel, com o monitor, listar ou alterar directamente o conteudo da memoria da segunda pagina. Recomenda-se neste caso, que se incluam nos programa rotinas temporarias de teste, feitas levando em conta as caracteristicas do monitor disponivel.

Nestes exemplos, sao utilizadas as rotinas acessiveis pela tabela de saltos. Qualquer duvida em relacao ao seu uso e particularidades, pode ser esclarecida pela lista do ultimo capitulo.

Nas rotinas em que ha interesse a devolucao do erro e feita em BC, de modo a que PRINT USR XXXXX ou LET A=USR XXXXX permitam um facil acesso do BASIC a este.

3.a - Comando de sistema.

Este tipo de comando e geralmente o mais facil de implementar, ja que movimenta poucos parametros. Sao os equivalentes aos comandos LIST*, CAT*, GOTO* Pathname, etc.. Todos sao executados de um modo muito semelhante e o programa apresentado pode ser facilmente alterado para executar qualquer um deles.

No exemplo optamos pelo CAT* que nos parece ser o que tem resultados mais espectaculares. Na primeira versao, a rotina apresenta a directoria corrente e na segunda o equivalente a CAT* "+.cod".

```
;
; TIMEX-SOFT
;
; ROTINA DE CAT PARA O FDD
;
; Retorna com o codigo de erro no acumulador (0 se tudo bem)
;
;
BUFDAT          EQU     2000H
BUFCOM          EQU     2100H
PAGEIN          EQU     0008H
PAGEOUT         EQU     0603H
PUTDAT          EQU     0605H
PUTCOM          EQU     0608H
RESPOSTA        EQU     0626H
;
DPYALL          EQU     11              ;display directoria corrente
DPYFILL         EQU     12              ;display ficheiros da directoria
                                        ;corrente de acordo com a matriz
;
; CAT* simples
;
CAT1            PUSH    IY              ;rotina de paginacao
                LD      IY,0
                CALL    PAGEIN
                POP     IY
;
```

```
                LD      A,DPYALL
                LD      (BUFCOM),A      ;codigo do comando
                CALL    PUTCOM          ;envio do comando
                CALL    RESPOSTA        ;fim do comando e escrita do texto
                LD      A,(BUFCOM+2)    ;codigo de erro
                LD      B,0
                LD      C,A             ;passagem do erro para o BASIC
                JP      PAGEOUT         ;retorno
;
  CAT* de todos os ficheiros de acordo com +.COD

CAT2            PUSH    IY
                LD      IY,0
                CALL    PAGEIN
                POP     IY
;
                LD      DE,BUFDAT       ;endereco do buffer de comunicacoes
                LD      HL,TEXTO        ;    "      da matriz dos ficheiros
                LD      BC,COMP         ;comprimento da matriz
                LDIR                    ;transfere o texto
                EX      DE,HL
                LD      (HL),0          ;os textos devem terminar c/ 0
                LD      A,COMP+1        ;o comprimento inclui o 0
                CALL    PUTDAT          ;envia a matriz
                LD      A,DPYFILL
                LD      (BUFCOM),A      ;codigo do comando
                CALL    PUTCOM          ;envio do comando
                CALL    RESPOSTA        ;fim do comando e escrita
                LD      A,(BUFCOM+2)    ;codigo de erro
                LD      B,0
                LD      C,A             ;passagem do erro para o BASIC
                JP      PAGEOUT         ;retorno
;
TEXTO           DEFM    "+.COD"
;
COMP            EQU     $-TEXTO
;
                END
```

Para alterar estas rotinas para os comandos seguintes, basta trocar os codigos de comando e escolher a primeira ou segunda versao, conforme tenham ou nao "Pathname".

```
DIM*    - CREATE
LIST*   - DPYSTK & DPYCDR
GOTO*   - GOTODR
GOSUB*  - CPUSH seguido de GOTODR
DRAW*   - CPOP
LIST*#  - DPYACH
FORMAT* - FORMATD ou CONFIG
CLOSE#* - CLOSALL
```

Podem ainda ser utilizados os comandos NEXTO e NEXTO, para os quais nao ha equivalencia no BASIC.

Para os comandos que requerem dois "Pathnames", como por exemplo LET*, basta colocá-los um a seguir ao outro em BUFDAT, separados por um 0 e envia-los como se fossem um so. Por exemplo para o comando equivalente a LET* "TESTE1" TO "TESTE2" e necessario enviar:

```
TESTE1 chr$(0) TESTE2 chr$(0)
```

O comprimento dos parametros para BUFDAT seria de 14.

3.b - SAVE e LOAD.

Estas funcoes talvez sejam as que tem interesse mais imediato para o programador. Sao executadas pelas rotinas SAVEP e LOADP e dado o numero de parametros necessarios para as definir, sao consideravelmente mais faceis de usar do que as anteriores.

A rotina SAVEP requer os parametros seguintes:

A - tipo de save 0 : programa
1 : array numerico.
2 : array de caracteres.
3 : codigo.
BC - numero de bytes a gravar.
DE - endereco inicial.
HL - numero da linha no caso de programa c/ LINE n.
A' - comprimento do nome em BUFDAT (incluido o 0 final).
(BUFDAT)- nome do ficheiro.

A rotina que se segue, grava no disco, na directoria corrente, o conteudo do ecran sob o nome de EXEMPLO.SCR.

```
  TIMEX-SOFT
;
; ROTINA DE SAVE DE ECRAN

; Retorna com o codigo de erro no acumulador (0 se tudo bem)
;

BUFDAT          EQU     2000H
BUFCON          EQU     2100H

PAGEIN          EQU     0008H
PAGEOUT         EQU     0603H
SAVEP           EQU     0620H

SCREEN          EQU     16384           ;endereco do ecran
SCRLEN          EQU     6912            ;comprimento em bytes do ecran

;
SAVESCR         PUSH    IY              ;paginacao
                LD      IY,0
                CALL    PAGEIN
                POP     IY

                LD      HL,TEXTO        ;ponteiro para o nome do ficheiro
                LD      DE,BUFDAT       ;buffer de comunicacoes
                LD      BC,COMP         ;comprimento do nome
                LDIR                    ;transferencia do nome
                EX      DE,HL
                LD      (HL),0          ;marca de fim de nome

                LD      A,COMP+1        ;comprimento do bloco a transf.
                EX      AF,AF           ;primeiro parametro
                LD      HL,0            ;nao e usado
                LD      DE,SCREEN
                LD      BC,SCRLEN
```

```
                LD      A,3                     ;tipo codigo
                CALL    SAVEP                   ;executa o comando
                LD      A,(BUFCOM+2)            ;codigo de erro
                LD      B,0
                LD      C,A                     ;passagem do erro para o BASIC
                JP      PAGEOUT                 ;retorno e paginacao
;
TEXTO           DEFM    "EXEMPLO.SCR"           ;nome do ficheiro
;
COMP            EQU     $-TEXTO
;
                END
```

Para segundo exemplo, vamos executar a funcao inversa usando a rotina LOADP. Esta rotina requer parametros diferentes:

(HEADER) - tipo de ficheiro pretendido.
(HEADER+1) - numero maximo de byte a carregar (se 0 e usado o comprimento total do ficheiro).
(HEADER+3) - endereco alternativo de carga (se 0 e usado o endereco que consta do ficheiro).
(BUFDAT) - nome do ficheiro.
B - comprimento do nome incluindo o 0 final.

```
;
; TIMEX-SOFT
;
; ROTINA DE LOAD DE ECRAN
;
  Retorna com o codigo de erro no acumulador (0 se tudo bem)
;
;
BUFDAT          EQU     2000H
BUFCOM          EQU     2100H
HEADER          EQU     214DH

LOADP           EQU     0523H
PAGEIN          EQU     0008H
PAGEOUT         EQU     0602H

;
LOADSCR         PUSH    IY                      ;Paginacao
                LD      IY,0
                CALL    PAGEIN
                POP     IY

                LD      HL,TEXTO                ;nome do ficheiro
                LD      DE,BUFDAT               ;buffer de comunicacoes
                LD      BC,COMP                 ;comprimento do nome
                LDIR
                EX      DE,HL
                LD      (HL),0                  ;marca de fim de nome

                LD      B,COMP+1                ;comprimento incluindo o 0
                LD      HL,0
                LD      (HEADER+1),HL           ;usar o valor de ficheiro
                LD      (HEADER+3),HL           ;idem
                LD      A,3                     ;tipo codigo
                LD      (HEADER),A
                CALL    LOADP                   ;executa o comando
                LD      A,(BUFCOM+2)            ;codigo de erro
```

```
                LD      B,0
                LD      C,A             ;passagem do erro para o BASIC
                JP      PAGEOUT         ;retorno e paginacao

TEXTO           DEFM    "EXEMPLO.SCR"   ;nome do ficheiro

COMP            EQU     $-TEXTO
;
                END
```

No caso de LOAD de um programa a ROM de extensão encarrega-se de liminar o programa anterior e instalar o novo. Para se executar um MERGE processo é ligeiramente mais complexo. A rotina seguinte faz o MERGE do ficheiro PROG.BAS.

```
;
; TIMEX-SOFT
;
; ROTINA DE MERGE DE PROGRAMAS
;
  Retorna com o codigo de erro no acumulador (0 se tudo bem)

BUFDAT          EQU     2000H
BUFCOM          EQU     2100H
CHAIN           EQU     2138H
HL_TMP          EQU     213FH
HEADER          EQU     214DH

LOADP           EQU     0623H
PAGEIN          EQU     0008H
PAGEOUT         EQU     0603H
;
;
MERGE           PUSH    IY              ;paginacao
                LD      IY,0
                CALL    PAGEIN
                POP     IY
;
                LD      HL,TEXTO        ;nome do ficheiro
                LD      DE,BUFDAT       ;buffer de comunicacoes
                LD      BC,COMP         ;comprimento do nome
                LDIR
                EX      DE,HL
                LD      (HL),0          ;marca de fim de nome
;
                LD      B,COMP+1        ;comprimento incluindo o 0
                LD      A,80H
                LD      (CHAIN),A       ;sinal de MERGE
                LD      A,0             ;tipo programa
                LD      (HEADER),A
                CALL    LOADP           ;executa o comando
                LD      A,(BUFCOM+2)    ;codigo de erro
                AND     A
                LD      B,0
                LD      C,A             ;passagem do erro para o BASIC
                JP      NZ,PAGEOUT      ;houve erro em LOADP
;
                LD      HL,(HL_TMP)     ;ponteiro p/ inicio do novo
                                        ;programa
                CALL    PAGEOUT         ;paginacao
```

```
                CALL    08DEH           ;rotina de merge do Spectrum
                LD      BC,0            ;nao houve erros
                RET                     ;fim da rotina
;
TEXTO           DEFM    "PROG.BAS"      ;nome do ficheiro
;
COMP            EQU     $-TEXTO
;
                END
```

3.c - Acesso directo ao conteudo do disco.

No sistema FDD, a informacao e gravada nos discos segundo o formato seguinte; 40 pistas, cada uma com 16 sectores, numerados de 0 a 15, cada um com 256 bytes. As primeiras 4 pistas, sao reservadas para o Sistema Operativo TOS e a quinta para a directoria. Nas unidades de disco com 1 Mb, a serem lancadas em breve, a estrutura e identica excepto que, existem 160 pistas com as pistas pares no lado 1 e as impares no lado 2.

Apesar dos sectores estarem numerados sequencialmente de 0 a 15, a informacao e guardada pelo TOS com um intervalo de 7 entre sectores consecutivos, de modo a melhorar o tempo de acesso. Por exemplo um ficheiro que ocupe 512 bytes e cujo primeiro sector e o numero 3, tem o segundo no sector 10.

O TOS permite ler e escrever directamente para os sectores do disco. Estas funcoes sao obviamente muito poderosas e devem ser usadas com as devidas precaucoes. A escrita no disco e irreversivel e no caso de erro, a informacao anterior fica permanentemente corrompida.

No exemplo escolhido o conteudo da pista da directoria e lido para um buffer em memoria. Esta rotina, por si so, nao e de grande interesse, uma vez que se pode obter o mesmo resultado em BASIC, fazendo INPUT do canal 0. No entanto, abre a possibilidade de escrita na directoria, o que pode permitir, por exemplo, criar um utilitario que recupere um ficheiro acidentalmente apagado, repondo correctamente a entrada na directoria.

A estrutura da directoria e compativel com a do sistema operativo CP/M, que ja se encontra instalado no FDD, e se espera seja em breve posto no mercado. Fica garantida, deste modo, a possibilidade de transferencia de ficheiros entre os dois sistemas operativos.

```
;
; TIMEX-SOFT
;
; ROTINA DE LEITURA DA DIRECTORIA DO TOS
;
; Retorna com o codigo de erro em A e BC (0 se tudo bem)
;
;
BUFDAT          EQU     2000H
BUFCOM          EQU     2100H
;
PUTDAT          EQU     0605H
PUTCOM          EQU     0608H
RESPOSTA        EQU     0626H
PAGEIN          EQU     0008H
PAGEOUT         EQU     0603H
;
RWSEC           EQU     27
;
;
RDDIR           PUSH    IY              ;paginacao
                LD      IY,0
```

```
                CALL    PAGEIN
                POP     IY
;
  leitura dos 16 sectores
;
                LD      DE,BUFFER       ;espaco para a directoria
                LD      B,0             ;contador
LOOP            CALL    RDSEC           ;leitura do sector
                LD      A,(BUFCOM+2)    ;codigo do erro
                AND     A
                JR      NZ,ERRO         ;erro na leitura
                PUSH    BC
                LD      HL,BUFDAT
                LD      BC,256          ;comprimento do sector
                LDIR                    ;transferir o sector
                POP     BC
                LD      A,B
                ADD     A,7             ;skew de 7
                AND     00001111B       ;so interessam os bits 0 -> 3
                LD      B,A
                JR      NZ,LOOP         ;ate ler todos os sectores
;
                XOR     A               ;nao houve erros
                LD      B,0
                LD      C,A             ;passagem do erro para o BASIC
ERRO            JP      PAGEOUT         ;paginacao e retorno

RDSEC           PUSH    DE
                PUSH    BC
                LD      E,B             ;numero do sector a ler
                LD      D,4             ;pista da directoria
                LD      C,0             ;disco A
                LD      A,RXSEC         ;codigo do comando
                LD      (BUFCOM),A
                CALL    PUTCOM          ;executar o comando
                CALL    RESPOSTA        ;esperar resultados
                POP     BC
                POP     DE
                RET
;
BUFFER          DEFS    4096            ;comprimento da pista
;
                END
```

3.d - Extensao do BASIC.

O FDD-BASIC deixa aberta ao utilizador a possibilidade de acrescentar comandos ao BASIC do Spectrum.

Nao existem limitacoes em relacao a sintaxe dos comandos criados pelo utilizador, desde que sigam o aspecto normal dos comandos em BASIC. No entanto, recomenda-se que seja usada uma sintaxe que nao entre em conflito com os outros comandos e preferencialmente que de origem a um erro do interpretador logo com o primeiro codigo. Por exemplo os caracteres acessiveis por "SYMBL SHIFT" sao uma boa escolha.

Para exemplificar esta possibilidade, apresentamos a seguir uma rotina que, depois de carregada em memoria atraves de:

LOAD * "COMANDO.NEW" CODE ADDR

; executada com o comando:

RANDOMISE USER ADDR

onde ADDR e um endereco compativel com a memoria disponivel, por exemplo
algures acima do RAMTOP, faz com que o Spectrum passe a aceitar, como
: ntacticamente correcto, o comando:

#

: e fica sendo equivalente a sequencia:

PRINT PAPER 0; INK 7; AT 10,8;" TESTE "

O comando apresentado nao inclui parametros. Para os usar e
ecessario prolongar a analise sintactica de modo a inclui-los e passar os
ilores para o modulo de execucao, o que pode ser feito do mesmo modo que
Spectrum o faz e usando as rotinas deste.

```
; TIMEX-SOFT

  ROTINA DE EXTENSAO DE BASIC
;
;
  Esta rotina e relocavel e tem 2 partes. A primeira prepara os vectores
, e a segunda executa o comando.
;


CBAS            EQU     051DH
'AGEOUT         EQU     0503H
 AGEIN          EQU     0008H
;
VECTOR          EQU     213BH
 H_ADD          EQU     5C5DH
,
CH_OPEN         EQU     1601H
 TMT_KEY        EQU     1B76H
 TMT_NEXT       EQU     1BF4H
;
'IVEIS          EQU     4

; Alteracao do vector de retorno do erro ao Spectrum
:
 OMMAND         LD      HL,COMP         ;comprimento da primeira parte
                ADD     HL,BC           ;apos USR XXXX BC tem o endereco de
                                        ;COMMAND e HL fica com
                                        ;o endereco absoluto da rotina

                PUSH    IY              ;paginacao
                LD      IY,0
                CALL    PAGEIN
                POP     IY
:
                LD      (VECTOR),HL     ;alterar a variavel de sistema
                JP      PAGEOUT         ;retorno ao BASIC
;
 OMP            EQU     $-COMMAND

; Ao chegar a RUNCOM o acumulador tem o codigo seguinte ao
```

```
; que originou o erro e esta selecionado o FDD-BASIC.
;
RUNCOM          LD      HL,(CH_ADD)     ;var. de sistema do Spectrum
                DEC     HL
                CP      0DH             ;fim da linha ?
                JR      Z,SEMI_OK
                CP      ":"             ;mais comandos na linha ?
                JR      Z,SEMI_OK
;
ERRO            INC     HL
                INC     HL              ;repoe o ponteiro
                LD      DE,11           ;local de retoma do processamento
                                        ;de erro
                PUSH    DE
                JP      PAGEOUT         ;paginacao e devolucao do erro
;
SEMI_OK         LD      A,(HL)
                CP      "#"             ;sera o nosso comando ?
                JR      NZ,ERRO

; A partir deste ponto a sintaxe esta correcta e falta repor o estado.
;
TUDO_OK         LD      HL,0
                ADD     HL,SP           ;repor o stack para
                LD      BC,NIVEIS       ;a posicao anterior ao Spectrum
                ADD     HL,BC           ;detectar o erro
                LD      SP,HL
;
; O Spectrum pode estar em modo de execucao ou analise sintactica.
; O teste e feito aqui.
;
                BIT     7,(IY+1)        ;teste do modo
                JR      NZ,RUN          ;executar o comando

; Estava em modo de analise sintactica.

                LD      HL,STMT_NEXT    ;reentrada no interpretador
                PUSH    HL
                JP      PAGEOUT

; Vai executar o comando.
;
UN              LD      A,2             ;canal de ecran
                CALL    CBAS
                WORD    CH_OPEN         ;abre canal
                LD      HL,COMANDOS     ;ponteiro para a sequencia
OOP             LD      A,(HL)
                CP      "#"             ;fim da sequencia
                JR      Z,FIM
                INC     HL
                PUSH    HL
                CALL    CBAS
                DEFW    PRINT_CH        ;rotina de escrita
                POP     HL
                JR      LOOP

IM              LD      HL,STMT_END     ;fim do comando
                PUSH    HL
                JP      PAGEOUT
;
```

```
COMANDOS        DEFB    17,0            ;PAPER 0
                DEFB    16,7            ;INK 7
                DEFB    22,10,8         ;AT 10,8
                DEFM    " TESTE "
                DEFB    "#"             ;fim da sequencia
;
                END
```

; .e - Processamento de ficheiros.

Todas as funcoes associadas ao processamento de ficheiros, sao tambem directamente acessiveis ao programador em ASSEMBLER.

Todo e qualquer acesso a ficheiros, para efeitos de escrita ou eitura, assume que o referido ficheiro foi previamente aberto. O modo de abertura vai condicionar o tipo de accoes possiveis sobre esse ficheiro.

Existem quatro modos de abertura de ficheiros:

1 - INPUT - Este modo permite unica e exclusivamente a leitura de dados do ficheiro. De cada vez que e feita uma leitura o ponteiro de ficheiro (file pointer) e actualizado para a posicao donde vai ser feita a proxima leitura. O comando de RESTORE reposiciona este ponteiro no inico do ficheiro.

2 - OUTPUT - Semelhante ao anterior, mas para escrita nos ficheiros. A utilizacao deste modo para ficheiros "nao nulos", bem como do comando RESTORE, destroi irremediavelmente o conteudo anterior destes.

- APPEND - Variante de modo de OUTPUT, na qual o ponteiro de ficheiro e inicializado para o fim do ficheiro corrente, de modo a permitir acrescentar dados ao seu conteudo.

4 - RANDOM - Modo de acesso directo, so possivel com tamanho de ficha definido (ver comprimento de fichas), e que permite a escrita ou leitura directa em qualquer ponto do ficheiro. Quando e escrita uma ficha, numa posicao para alem do fim de ficheiro, o TOS aumenta automaticamente o tamanho do ficheiro ate a incluir. No caso de uma leitura nestas condicoes, e devolvido o

A rotina de exemplo escolhida para este paragrafo e muito simples, retende apenas demonstrar os mecanismos basicos de abertura e acesso a icheiros. Assume-se que existe um ficheiro, onde cada ficha tem um byte dedicado a indicar o tipo de ficha, e que quando esse byte e igual a 0FFh ntao a ficha foi anulada. A rotina faz a compactacao do ficheiro etirando as fichas anuladas.

```
; TIMEX-SOFT
;
  ROTINA DE COMPACTACAO DE FICHEIROS
;
```

```
  PARAMETROS DE ENTRADA:
;
;               HL      - Ponteiro para o nome do ficheiro (terminado c/ 0)
                DE      - Numero de bytes de cada ficha
;
; PARAMETROS DE SAIDA:
;
                A       - Codigo de erro (0 se tudo bem)
;               BC      - Novo numero de fichas no ficheiro
;
  Assume-se que o byte que indica o tipo e o primeiro da ficha.
;
;
 UFDAT          EQU     2000H
_UFCOM          EQU     2100H
TCHAN           EQU     212EH

 UTDAT          EQU     0605H
PUTCOM          EQU     0600H
PESPOSTA        EQU     0626H
 AGEIN          EQU     0008H
PAGEOUT         EQU     0603H
;
 PENF           EQU     0
_LOSEF          EQU     1
WRITEF          EQU     15
 EADF           EQU     16

;
OMPACT          LD      (COMP),DE       ;guardar comprimento de ficha
                PUSH    HL
;
                PUSH    IY              ;paginacao
                LD      IY,0
                CALL    PAGEIN
                POP     IY

  transferir nome do ficheiro para o buffer de dados
;
                POP     HL
                LD      DE,BUFDAT
                LD      B,1
OPY             LD      A,(HL)
                LD      (DE),A
                AND     A
                JR      Z,ENDCOPY       ;o nome termina com 0
                INC     HL
                INC     DE
                INC     B
                JR      COPY

NDCOPY          LD      A,B             ;comprimento do nome
                CALL    PUTDAT

                LD      D,00000011B     ;modo "random"
                LD      E,0             ;records
                LD      IX,(COMP)       ;comprimento do record
                LD      A,OPENF         ;comando de abertura de ficheiro
                LD      (BUFCOM),A
                LD      A,1             ;numero do canal a usar
                LD      (TCHAN),A
                CALL    PUTCOM          ;envio do comando
```

```
                CALL    RESPOSTA        ;aguardar fim do comando
                LD      A,(BUFCOM+2)    ;le codigo de erro
                AND     A
                JP      NZ,PAGEOUT      ;retorna caso haja erro
;
; o ficheiro esta pronto a ser usado
;
                LD      HL,1            ;ponteiro para ficha a ler
                LD      DE,1            ;ponteiro para ficha a escrever
READLP          CALL    READ            ;le ficha numero <HL>
                JR      NZ,READEND      ;salta quando ha erro
                                        ;normal ao chegar ao fim do ficheiro
                CALL    WRITE           ;escreve a ficha em <DE>
                LD      A,(BUFFER)      ;primeiro byte da ficha lida
                CP      0FFH            ;anulada ?
                JR      Z,ANULADA
                INC     DE              ;proxima ficha
ANULADA         INC     HL              ;idem
                JR      READLP          ;ate ao fim do ficheiro
;
; a fase de compactacao terminou, falta fechar o ficheiro
;
READEND         PUSH    DE              ;numero de fichas activas
                LD      A,CLOSEF        ;comando de fecho de ficheiro
                LD      (BUFCOM),A
                LD      A,1             ;numero do canal
                CALL    PUTCOM          ;executa o comando
                CALL    RESPOSTA        ;aguarda fim de comando
                LD      A,(BUFCOM+2)    ;codigo de erro
                POP     BC              ;parametro a devolver
                JP      PAGEOUT
;
;  Ler a ficha com o numero em HL para o buffer
;
READ            PUSH    HL
                PUSH    DE
                EX      DE,HL           ;o numero do record vai em DE
                LD      B,1             ;numero do canal
                LD      C,0             ;MSB do numero de record
                LD      A,READF
                LD      (BUFCOM),A
                CALL    PUTCOM          ;envia o comando
                CALL    RESPOSTA
                LD      BC,(COMP)
                LD      HL,BUFDAT
                LD      DE,BUFFER
                LDIR                    ;tranferir a ficha
                POP     DE
                POP     HL
                LD      A,(BUFCOM+2)    ;codigo de erro
                AND     A
                RET
;
; Escrever a ficha em buffer para o registo DE no ficheiro
;
WRITE           PUSH    HL
                PUSH    DE
                LD      BC,(COMP)
                LD      DE,BUFDAT
                LD      HL,BUFFER
                LDIR                    ;transferir a ficha para BUFDAT
                LD      A,(COMP)        ;comprimento da ficha ( 0<x<255 )
```

```
                CALL    PUTDAT          ;enviar ficha para o TOS
                LD      C,0
                LD      B,1             ;numero do canal
                POP     DE              ;numero do record
                PUSH    DE
                LD      A,WRITEF        ;comando de escrita em ficheiro
                LD      (BUFCOM),A
                CALL    PUTCOM          ;executar o comando
                CALL    RESPOSTA        ;aguardar fim de comando
                POP     DE
                POP     HL
                RET

  variaveis
;
COMP            DEFS    2               ;comprimento da ficha
UFFER           DEFS    256             ;local temporario p/ as fichas
;
                END
```

- LISTA DE REFERENCIA.

Neste capitulo sao descritas todas as rotinas de extensao de BASIC, cessiveis pela tabela de saltos, e todas as funcoes do TOS.

.a - Rotinas do FDD-BASIC.

UTDAT — Envia um bloco de dados para o TOS com o comprimento maximo de 256 bytes.

TILIZACAO — Sempre que e necessario enviar para o TOS Pathnames, texto ou codigo, este e colocado em BUFDAT e a rotina invocada para fazer a transmissao. O TOS nao confirma a recepcao de dados, limita-se a recebe-los e assume que a operacao a realizar sobre eles sera descrita pelo proximo bloco de comando. Ao fazer chamadas a esta rotina sem executar um comando os novos dados sobrepoem-se aos anteriores sem qualquer efeito particular.

ARAMETROS — A - comprimento do bloco de dados.
(BUFDAT...BUFDAT+0FFh) - dados a enviar.

REG.ALTERADOS — AF

ESULTADOS — HL - devolvido com BUFDAT.
(COMP) - comprimento do bloco transmitido.
(TIPO) - identificacao do bloco - 0D0h.

NOTAS — Tenta indefinidamente o envio ate conseguir, ou o operador fazer "BREAK", caso em que o controle e devolvido ao Spectrum pela rotina SYNERR (ver fim do capitulo).

UTCOM — Envia um bloco de comando para o TOS.

TILIZACAO — Rotina usada para o envio dos comandos para o TOS e desencadear a sua execucao. Apos enviar os dados pertinentes a um comando com PUTDAT, esta rotina transmite ao TOS o comando pretendido e os parametros necessarios. Depois de executada o TOS responde com o resultado do comando. Os parametros a transmitir ao TOS sao passados pelos registos internos do processador.

ARAMETROS — (BUFCOM) - codigo do comando a executar.

EG.ALTERADOS — AF

RESULTADOS — HL - devolvido com BUFCOM.
13 primeiros bytes de BUFCOM c/ o codigo do comando e o conteudo dos registos AF, BC, DE, HL, IX e IY.
(COMP) - alterado para 13.
(TIPO) - identificacao do bloco - 0C0h.

NOTAS — Tenta indefinidamente o envio ate conseguir ou o operador fazer "BREAK", caso em que o controle e devolvido ao

Spectrum pela rotina SYNERR. Em BUFCOM ficam, por ordem, os seguintes valores: codigo do comando, AF, BC, DE, HL, IX e IY.

GETBLOCK — Recebe do TOS um bloco de dados ou comando.

UTILIZACAO — Conjuntamente com PUTDAT e PUTCOM completa o grupo de rotinas indispensaveis para a execucao de todos os comandos do TOS. Apos o envio de um bloco de comando com PUTCOM o TOS tenta executar o comando pedido e gera uma resposta contendo, quando necessario, uma mensagem de erro ou um pedido de actuacao. Comandos que devolvem informacao, podem faze-lo de dois modos: quando e possivel ela e devolvida nos registos do processador, quando e muito extensa e nao cabe nos registos o TOS envia primeiro um bloco de dados e depois a informacao de fim de comando, de modo que em certos comandos e necessario chamar GETBLOCK ate receber indicacao de fim de comando (ver RESPOSTA).

PARAMETROS — Nenhum

RESULTADOS — Carry set -> recebeu comando e todos os registos tem os valores devolvidos pelo TOS. Os registos que nao sao usados pelo TOS ficam com o valor que tinham quando do ultimo PUTCOM.
Carry reset -> recebeu bloco de dados. Comprimento do bloco em COMP e dados a partir de BUFDAT.

NOTAS — Espera indefinidamente por uma transmissao ou ate o operador fazer "BREAK", caso em que o controle e devolvido ao Spectrum pela rotina SYNERR.

SENDBL — Envia um bloco para o TOS.

UTILIZACAO — Quando a aplicacao requer controle absoluto sobre o modo como se processam as comunicacoes deve ser usada esta rotina em vez de PUTDAT ou PUTCOM. As rotinas anteriores fazem uso desta para os efeitos de transmissao. Como nao testa o comando de "BREAK" o controle do programa nunca e perdido. O bloco a transmitir deve estar de acordo com os valores das variaveis COMP e TIPO.

PARAMETROS — HL - ponteiro para o bloco.
(COMP) - comprimento do bloco.
(TIPO) - tipo de bloco.
(CONT) - numero de tentativas de transmissao - sempre 10.

RESULTADOS — AF - Zero set quando bem sucedida.

NOTAS — Tenta dez vezes o envio do bloco, caso nao consiga devolve a flag Zero reset.

GETBL — Recebe um bloco do TOS.

UTILIZACAO — Funcao inversa da anterior e substitui GETBLOCK nas mesmas

semelhantes.

PARAMETROS — nenhum.

RESULTADOS — AF - Zero set quando bem sucedida.
(COMP) - comprimento do bloco recebido.
(TIPO) - tipo do bloco 00h -> dados.
01h -> comando.
Bloco recebido em BUFDAT ou BUFCOM, conforme o tipo.

NOTAS — Tenta dez vezes receber um bloco, caso nao consiga devolve a flag Zero reset. Os blocos do tipo comando tem sempre 45 bytes de comprimento, sendo os primeiros 13 o codigo do comando e os registos pela ordem AF, BC, DE, HL, IX e IY. Os ultimos 32 bytes tmazem o texto associado ao erro havido, ou a mensagem anterior, caso nao haja erro.

SAVEP — Rotina que executa funcoes similares ao SAVE do BASIC. Permite guardar num ficheiro, ou enviar para um SCP com formato de SAVE, um programa, o conteudo de uma zona de memoria, um ecran ou o conteudo de uma variavel.

UTILIZACAO — Se bem que nao seja indispensavel, esta rotina facilita ao programador a realizacao de acoes do tipo SAVE e pode ser usada quando o teste do comando de "BREAK" e tolerado. Quando esta condicao nao e aceitavel estas funcoes devem ser implementadas a custa das rotinas SENBL e GETBL.

PARAMETROS — A - tipo de ficheiro
0 - programa
1 - matriz numerica
2 - matriz de caracteres
3 - imagem de memoria ou ecran
A' - comprimento do nome do ficheiro.
(BUFDAT...) - nome do ficheiro ou SCP terminado c/ 00h.
HL - numero de linha para arranque automatico de programas, 0 quando nao e necessario.
DE - endereco do inicio do codigo.
BC - comprimento do codigo.

RESULTADOS — AF - codigo de erro - 0 e flag set se tudo bem.
(BUFCOM+13...) - mensagem de erro, quando seja o caso.
Restantes registos tomam valores indefinidos.

NOTAS — A indicacao do tipo de ficheiro nao dispensa o uso de HL e DE para passagem de parametros. Esta rotina gera um cabecalho para o ficheiro com toda a informacao necessaria para a identificacao pelo BASIC. O cabecalho tem sete bytes no caso de um programa e cinco bytes nos outros casos. Sao respectivamente: o tipo de ficheiro, o numero de linha para arranque automatico (so em programas), comprimento do codigo e endereco de carga.

LOADP — Rotina que executa funcoes semelhantes ao LOAD do BASIC. Permite carregar em memoria, um programa, um ficheiro de codigo, um ecran ou o valor de uma variavel, provenientes de um ficheiro ou SCP.

UTILIZACAO — Aplicam-se aqui as mesmas consideracoes que para a rotina

anterior.

'RAMETROS (HEADER) - tipo do ficheiro
0 - programa
1 - matriz numerica
2 - matriz de caracteres
3 - codigo ou ecran
(HEADER+1,2) - numero de bytes a ler, 0 -> ficheiro completo
(HEADER+3,4) - endereco de carregamento, 0 -> endereco que consta do ficheiro.
(BUFDAT...) - nome do ficheiro ou SCP.

RESULTADOS AF - codigo de erro - 0 e flag set se tudo bem.
(BUFCON+13...) - mensagem de erro, quando seja o caso.
Restantes registos tomam valores indefinidos.

'OTAS No caso de carregamento de programas so e necessario o primeiro parametro, ecarregando-se a rotina de eliminar o programa corrente e instalar o novo. A variavel CHAIN controla o modo como e feita esta operacao, executando o equivalente a um LOAD quando igual a 0 e um MERGE nos outros casos (ver exemplo de MERGE).

ARTMEM Escreve num ficheiro o conteudo de uma zona de memoria.

TILIZACAO Esta e as duas rotinas que se seguem correspondem a um nivel mais baixo das duas anteriores, ou seja, sao usadas por estas para a execucao do comando, mas podem ser uteis numa aplicacao mais especifica. Esta e particularmente util para a escrita de blocos, de qualquer comprimento, em ficheiros sequenciais de codigo ou texto.

ARAMETROS (TCHAN) - numero do canal aberto para o ficheiro.
(PROLEN) - numero de bytes a escrever (2 byte).
(PROSTR) - endereco do inicio do codigo (2 byte).

RESULTADOS AF - codigo de erro.
(BUFCON+13...) - mensagem de erro, quando seja o caso.

.OTAS O ficheiro a usar deve ser previamente aberto em modo OUTPUT, sem comprimento de record definido. A rotina nao fecha o ficheiro.

:DMEM Carrega em memoria o conteudo de um ficheiro.

UTILIZACAO Rotina inversa da anterior, util para a leitura de ficheiros sequenciais.

PARAMETROS (TCHAN) - numero do canal aberto para o ficheiro.
(PROLEN) - numero de bytes a carregar.
(PROSTR) - endereco de carregamento.

'ESULTADOS AF - codigo de erro.
(BUFCON+13...) - mensagem de erro, quando seja o caso.

NOTAS O ficheiro a usar deve ser previamente aberto em modo INPUT, sem comprimento de record definido. A rotina nao fecha o ficheiro.

RDBLOC — Le de um ficheiro para BUFDAT um bloco com ate 256 bytes.

UTILIZACAO — Rotina de leitura muito especializada, indicada para a exploracao de ficheiros sequenciais de texto.

PARAMETROS — (TCHAN) - numero do canal aberto para o ficheiro.
DE - numero de bytes a ler.
C - modo 0FFH - modo identico ao INPUT do BASIC
<>0FFH - leitura ate ao caracter em C

RESULTADOS — AF - codigo de erro.
(BUFDAT...) - bloco lido.
(BUFCON+13...) - mensagem de erro, quando seja o caso.
HL - alterado para BUFDAT.
BC - numero de bytes realmente lidos.

NOTAS — O ficheiro a usar deve ser previamente aberto em modo INPUT, sem comprimento de record definido. A rotina nao fecha o ficheiro.

CBAS — Executa rotinas da ROM do Spectrum estando a ROM de expansao seleccionada.

UTILIZACAO — Ao utilizar os comandos do TOS e por vezes necessario recorrer a rotinas da ROM do Spectrum. Esta rotina apresenta um modo facil de o fazer, preservando todos os registos e executando todo o mecanismo de paginacao.

PARAMETROS — Endereco da rotina a executar nos dois bytes seguintes a instrucao CALL CBAS, com o byte de menor peso primeiro.

REG.USADOS — Zero reset.

NOTAS — A rotina executada na ROM do Spectrum deve terminar com RET e o stack na mesma posicao em que o recebeu. Esta rotina nao deve ser usada para devolver o controlo ao Spectrum.

RESPOSTA — Rotina de terminacao dos comandos de disco.

UTILIZACAO — Esta rotina executa todas as chamadas a GETBLOCK necessarias para a terminacao correcta dos comandos, e assume o controle do programa, no caso do comando usar o ecran ou requerer parametros directamente do utilizador. Caso haja problemas com as rotinas de comunicacoes (ligacao interrompida ou controlador desligado), a tecla de BREAK e testada e a rotina pode abortar, devolvendo o comando ao Spectrum. Esta situacao e normalmente fatal. Usar esta rotina constitui a maneira mais facil do utilizador terminar os comandos do TOS.

PARAMETROS — Nenhum.

RESULTADOS — (BUFCON+2) - codigo de erro.
(BUFCON+13,BUFCON+45) - mensagem de erro, quando e o caso.

.b - Rotina SYNERR.

Quando ha uma quebra nas comunicacoes entre o Spectrum e o TOS, e-
eito um teste do comando de "BREAK", e eventualmente o comando cortado e
o controle devolvido ao Spectrum por intermedio desta rotina.

```
SYNERR      LD      SP,(ERR_SP)     ;var. do Spectrum
            LD      (IY+0),A        ;transfere o codigo de erro
            LD      HL,(CH_ADD)     ;ponteiro p/ o codigo que originou
                                    ;o erro
            LD      (X_PTR),HL      ;var. do Spectrum
            LD      HL,SET_STK      ;rotina de reposicao do stack e
                                    ;limpeza do "workspace" do Spectrum
            PUSH    HL
            JP      0603H           ;paginacao e salto indirecto
```

4.c - Codigos utilizados pelo TOS e FDD-BASIC.

```
  Codigos recebidos em (BUFCOM) apos GETBLOCK
;
ENDCMD      EQU     80H     ;fim de comando, enviado p/ confirmar
                            ;a sua execucao.
WRITEL      EQU     81H     ;pedido do TOS de escrita no ecran do texto
                            ;presente em BUFDAT.
WRWAIT      EQU     82H     ;pedido de escrita de texto e confirmacao
                            ;por <Enter>.
RDLTER      EQU     83H     ;pedido de escrita do texto e devolucao da
                            ;proxima tecla premida.
;
; Apos um pedido do TOS o valor e devolvido em A
; e DONE colocado em (BUFCOM) para confirmar a satisfacao do pedido
;
DONE        EQU     91H     ;confirma execucao do pedido.
;
; Tipos de blocos identificados na var. TIPO
;
DATA        EQU     00H     ;tipo dados
CMD         EQU     00H     ;tipo comando
;


;
; Codigos das funcoes disponiveis do TOS
;
;
OPENF       EQU     0       ;abrir ficheiro
                            ;Pathname previamente enviado por PUTDAT
                            ;D - modo
                            ;       0. append
                            ;       1. input
                            ;       2. output
                            ;       3. random
                            ;E - tipo de ficheiro
                            ;       0. record
```

```
                                ;         1. stream
                                ;A - numero de canal
                                ;IX - comprimento de record

CLOSEF          EQU     1       ;fecha ficheiro
                                ;A - numero do canal

CREATE          EQU     2       ;cria ficheiro
                                ;Pathname previamente enviado por PUTDAT
                                ;A - tipo
                                ;         0. ficheiro ou directoria
                                ;         1. apenas ficheiros

DPYSTK          EQU     3       ;mostra stack de directorias
                                ;envia pedidos de escrita

ERASEF          EQU     4       ;apaga ficheiro
                                ;Pathname previamente enviado por PUTDAT
                                ;A - confirmacao
                                ;         0. requer confirmacao
                                ;         1. executa imediatamente
                                ;pode enviar pedidos de escrita

MOVEF           EQU     5       ;muda nome de ficheiros, directorias ou SCP's
                                ;Pathnames previamente enviados por PUTDAT

COPYF           EQU     6       ;copia um fich./SCP para um fich./SCP
                                ;Pathnames previamente enviados por PUTDAT
                                ;pode enviar pedidos de escrita
;
GOTODR          EQU     7       ;torna corrente a directoria indicada
                                ;Pathname ou drive enviado por PUTDAT
                                ;A - modo
                                ;         0. c/ Pathname
                                ;         1. c/ nome de drive
;
DPYCDR          EQU     8       ;mostra directoria corrente
                                ;pode enviar pedidos de escrita
;
DPOP            EQU     9       ;torna corrente a ultima directoria do stack
;
DPUSH           EQU     10      ;poe a directoria corrente p/ o topo do stack
;
DPYALL          EQU     11      ;DAT de todos os fich. da dir. corrente
                                ;envia pedidos de escrita
;
DPYFIL          EQU     12      ;DAT dos ficheiros de acordo c/ template
                                ;Pathname previamente enviado por PUTDAT
                                ;envia pedidos de escrita
;
DPYCHN          EQU     13      ;mostra informacao sobre um canal
                                ;B - numero do canal
                                ;envia pedido de escrita
;
DPYACH          EQU     14      ;mostra inf. sobre todos os canais abertos
                                ;envia pedido de escrita
;
WRITEF          EQU     15      ;escreve em ficheiros ou SCP's
                                ;B - numero do canal
                                ;--ficheiro aberto em modo de records--
                                ;C DE - numero do rec. (0 -> rec.seguinte)
                                ;--ficheiro aberto em modo stream--
```

```
                                ;DE<>0 - numero de bytes a escrever
                                ;C=0   escrita ate ao car. em C ou 256 bytes
                                ;devolve em DE o numero de bytes escritos

READF           EQU     15      ;le ficheiros ou SCP's
                                ;B - numero do canal
                                ;--ficheiro aberto em modo de records--
                                ;C DE - numero do rec. (0 -> rec.seguinte)
                                ;--ficheiro aberto em modo stream--
                                ;DE<>0 - numero de bytes a ler ( <256 )
                                ;DE=0  C=0FFH - modo igual ao INPUT de BASIC
                                ;      C<>FFH - leitura ate o car. em C inc.
                                ;devolve em DE o numero de bytes lidos

FORMATD         EQU     17      ;formata diskettes e instala o O.S.
                                ;Pathname e drive enviado por PUTDAT
                                ;A - tipo
                                ;       0. 40 pistas
                                ;     <>0. 80 pistas

FREECH          EQU     18      ;pedido do numero de um canal livre
                                ;B - numero do canal disponivel

;
CHSTAT          EQU     19      ;pedido de inf. detalhada s/ um canal
                                ;FCB devolvido em BUFCOM

;
PROTUP          EQU     20      ;define atributos do ficheiro
                                ;Pathname previamente enviado por PUTDAT
                                ;A - modo
                                ;       0. desprotege
                                ;       1. protege
                                ;       2. visivel
                                ;       3. invisivel

;
CONFIG          EQU     21      ;configuracao de um SCP
                                ;Pathname previamente enviado por PUTDAT
                                ;envia pedidos de escrita

;
RSTORE          EQU     22      ;coloca o ponteiro de fich. no inicio deste
                                ;H - numero do canal

;
GETVER          EQU     23      ;devolve numero de versao do TOS em HL

;
XSUB            EQU     24      ;executa subrotina no TOS
                                ;IY - endereco da rotina a executar
                                ;todos os outros registos sao passados
                                ;para a rotina.
                                ;Todos os registos sao devolvidos como a
                                ;rotina os deixa.

;
RXDAT           EQU     25      ;recebe bloco de memoria do TOS
                                ;A - numero de bytes a ler
                                ;DE - endereco do bloco

;
TXDAT           EQU     26      ;transmite bloco para a memoria do TOS
                                ;bloco previamente enviado por PUTDAT
                                ;DE - endereco de carga do bloco

;
RXSEC           EQU     27      ;le sector do disco
                                ;C - numero do drive ( 0-3 )
                                ;D - numero de pista
                                ;E - numero do sector
```

```
WSEC        EQU     28      ;escreve sector no disco
                            ;sector previamente enviado por PUTDAT
                            ;C - numero do drive ( 0-3 )
                            ;D - numero da pista
                            ;E - numero do sector

LOSALL      EQU     29      ;fecha todos os canais
;
NEXT1       EQU     31      ;cria directoria alternativa para as funcoes
                            ;de LOAD e MERGE
                            ;Pathname previamente enviado por PUTDAT
;
NEXT2       EQU     32      ;retira a directoria alternativa
```

4.d - Variaveis de sistema do FDD-BASIC.

Todas as variaveis mencionadas abaixo se encontram na RAM do "INTERFACE" e para ter acesso a elas e necessario que a pagina do FDD-BASIC esteja selecionada.

A memoria esta disponivel para o utilizador de 2155H ate 23FFH.

Os enderecos nao indicados na tabela correspondem a variaveis internas do sistema cuja utilizacao e de interesse remoto para o utilizador.

```
BUFDAT      EQU     2000H   ;buffer de dados para as rot. de comunicacao
BUFCOM      EQU     2100H   ;buffer de comandos para as rot. de comun.
CHAN        EQU     212EH   ;numero de canal
TIPO        EQU     212FH   ;tipo de bloco
COMP        EQU     2130H   ;comprimento do bloco
CHAIN       EQU     2132H   ;sinal de LOAD ou MERGE
MODO        EQU     2133H   ;modo de fim de input (ver READF e WRITEF)
VECTOR      EQU     2138H   ;endereco da rot. de retorno de erro ao Sp.
ABORT       EQU     213DH   ;endereco da rot. de retorno ao Spectrum,
                            ;apos executar um comando
HL_TMP      EQU     213FH   ;local temp. para HL (util p/ MERGE)
PROLEN      EQU     2145H   ;comprimento do codigo
PROSTR      EQU     2147H   ;endereco de inicio do codigo
HEADER      EQU     214DH   ;buffer para passagem de parametros
```

4.e - Bibliografia.

Manual de utilizacao do sistema FDD - TIMEX
Complete Spectrum ROM disassembly - Dr. Ian Logan
Z80 Assembly Language Programming - Lance A. Leventhal